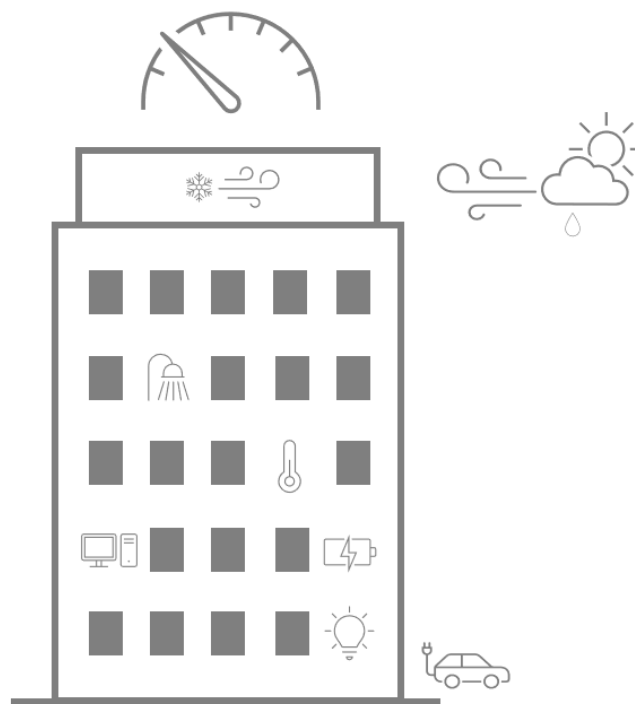


FMH606 Master's Thesis 2023
Industrial IT and Automation

Development of Predictive Machine Learning Algorithm for Energy Usage in Buildings



Øystein Guldberg

Faculty of Technology, Natural sciences and Maritime Sciences
Campus Porsgrunn

Course: FMH606 Master's Thesis, 2023

Title: Development of Predictive Machine Learning Algorithm for Energy Usage in Buildings

Number of pages: 85

Keywords: Machine learning, predictive energy model, energy usage, energy forecasting in buildings

Student: Øystein Guldberg

Supervisor: Anthoula Mountzouri, Carlos Pfeiffer

External partner: Yodiwo

Summary:

A model for energy usage forecasting can be a tool to help reduce the energy consumed by buildings, and thereby also reduce both the energy costs and carbon footprint of buildings.

In order to create a good prediction model, the inputs containing valuable information are found by data analysis. The data analysis mainly uncovers that the training data in this project contains certain cyclical patterns for daily, weekly, and yearly variations. And that one of the most important parameters influencing changes in energy consumption is cooling caused by the outdoor temperature.

This thesis investigates using two different machine learning algorithms to create energy consumption models for daily and hourly energy predictions. The first model type is long-short term memory (LSTM) and the second is gradient boosted machines. Which model type produces the best results will depend on the building, and the desired prediction horizon which may be the next hour, day, or month.

The data used for model training in the project is recorded from two separate buildings located in Athens, Greece. Where the best predicting model for building #1 is a LSTM model with a MAE of 608 kWh (about 15% of test set mean), and for building #2 a gradient boosting model with a MAE of 1208 kWh (about 8% of test set mean).

Preface

This report is written as part of the master thesis in the eighth semester of the online master Industrial IT and Automation program (IIA) at the University of Southeast Norway (USN).

The work performed during this thesis is part of a collaboration between USN and the external partner Yodiwo. Yodiwo is a company helping customers optimize a variety of tasks within the domain of building and space management by utilizing IoT, AI and other digital tools. In this thesis, the Yodiwo branch located in Athens has been the contributor of data and knowledge.

For the performance and support during this thesis I would like to thank Yodiwo and in particular Anthoula for providing knowledge, guidance, and constant support during the work. I would also like to direct a special thanks to Carlos for his support and expertise in the field of modelling and engineering work.

This project relies on implementations in Python, and depends on important libraries such as XGBoost, TensorFlow and Keras in particular. Plots in the results part are all created with Python. All other figures and illustrations are created by the author with the help of Microsoft Office tools.

Fundamental knowledge in the fields of machine learning, model development, programming and general thermal dynamics will help the interested readers enjoy this report. The work performed during this thesis could be useful for other peers in the field of data analysis and machine learning community, or people interested in energy usage in buildings in general.

Full source code developed during the thesis can be accessed through these links: [Data import](#), [Data analysis](#), [XGBoost model](#), [LSTM model](#).

“If I have seen further, it is by standing on the shoulders of Giants.”

- Isaac Newton

Porsgrunn, 2023.05.15

Øystein Guldberg

Contents

Preface3

Contents.....4

Abbreviations6

1 Introduction7

 1.1 Background information7

 1.2 Energy and Thermal Models8

 1.2.1 *White box model*8

 1.2.2 *Black box model*.....8

 1.2.3 *Grey box model*8

 1.2.4 *Important parameters*8

 1.2.5 *Related work*.....10

 1.3 Project objectives10

 1.4 Report structure.....10

2 Methodology11

 2.1 Artificial Intelligence and Machine Learning11

 2.1.1 *Timeseries and Machine Learning*13

 2.1.2 *Training the ML algorithm*13

 2.2 Neural Network - NN14

 2.2.1 *Training the neural network*.....16

 2.2.2 *Recurrent Neural Network (RNN)*18

 2.2.3 *Long Short-term memory (LSTM)*.....20

 2.2.4 *Deep learning*.....21

 2.3 Decision Tree - DT22

 2.3.1 *Random forest*.....22

 2.3.2 *Boosting*.....23

 2.3.3 *Gradient boosting machines*.....23

 2.4 Python23

 2.4.1 *Google Colaboratory – Colab*24

 2.4.2 *TensorFlow and Keras*.....24

 2.4.3 *XGBoost*.....24

 2.4.4 *Scikit-learn*25

 2.4.5 *Pandas*.....25

 2.4.6 *Matplotlib*25

 2.5 Data preparation26

 2.5.1 *Raw data format*28

 2.5.2 *Pre-processing and aggregation of data*29

 2.5.3 *Time stamps*32

 2.6 Feature Engineering32

 2.6.1 *Weather data*.....33

 2.6.2 *Energy consumption*33

 2.6.3 *Time Features*.....34

 2.6.4 *Lag Features*.....35

 2.7 Data Analysis35

 2.7.1 *Trend*35

 2.8 Modelling36

 2.8.1 *Extreme gradient boosting*.....37

 2.8.2 *LSTM*.....37

2.8.3	<i>Validation of models</i>	39
2.8.4	<i>Data standardization</i>	39
2.8.5	<i>One-hot encoding</i>	39
2.8.6	<i>Outlier handling</i>	39
3	Data analysis results	40
3.1	Weather data analysis	40
3.2	Energy data analysis	41
3.2.1	<i>Building #1</i>	41
3.2.2	<i>Building #2</i>	42
3.2.3	<i>Both buildings</i>	43
3.2.4	<i>Trend</i>	44
3.3	Combined data analysis	46
3.3.1	<i>Timely energy features</i>	46
3.3.2	<i>Correlation</i>	49
3.3.3	<i>Autocorrelation</i>	51
3.4	New temperature feature	53
4	Modelling results	55
4.1	Outlier handling	55
4.1.1	<i>Standard deviations From grouped Mean (SFM)</i>	55
4.1.2	<i>Temperature scatter</i>	56
4.2	Baseline model	57
4.3	Extreme gradient boosting	58
4.3.1	<i>Timely features – daily energy consumption</i>	58
4.3.2	<i>All features – daily energy consumption</i>	60
4.3.3	<i>All features – hourly energy consumption</i>	62
4.3.4	<i>Model parameter importance</i>	63
4.4	LSTM models	64
4.4.1	<i>Single input model</i>	64
4.4.2	<i>General model</i>	66
4.5	Future predictions	68
5	Discussion	69
5.1	Pre-processing	69
5.2	Data Analysis	69
5.2.1	<i>Outlier detection</i>	70
5.3	Models	71
5.3.1	<i>Model validation</i>	71
5.3.2	<i>Gradient Boosting Machines</i>	71
5.3.3	<i>LSTM</i>	72
5.3.4	<i>Overall model results</i>	73
5.4	Future work	74
6	Conclusion	75
	References	76
	Appendices	79

Abbreviations

ANN – Artificial Neural Network

NN – Neural Network

ML – Machine Learning

AI – Artificial Intelligence

SVM – Support Vector Machine

LSTM – Long Short-Term Memory

GRU – Gradient Recurrent Unit

HVAC – Heating, Ventilation and Air Conditioning

CSV – Comma-Separated Values file

JSON – JavaScript Object Notation

UTC – Coordinated Universal Time (Universal Time Coordinated)

1 Introduction

This introduction chapter discloses the purpose and goal of the project in the contexts with some of the background information, alongside an introduction to energy and thermal models in general. At the end of the chapter, there is also information about the report structure.

1.1 Background information

If the world shall reach the goal of limiting global warming and climate change by reducing the total emission of greenhouse gasses in accordance with the Paris climate agreement, every sector must contribute. Buildings consume nearly 40% of the overall energy worldwide and are thus responsible for the corresponding carbon footprint [1]. It is estimated that this portion is larger than both the industry sector (about 32% of total) and the transport sector (about 28% of total) [2]. It is even estimated that the energy demand in buildings will increase mainly because of population growth, increased number of buildings and amount of floor space. All of which underlines the importance of good energy efficiency in the building sector.

Of the total energy consumed by the building sector, a study from 2018 shows that operation of residential buildings accounts for about 61% and operation of non-residential buildings accounts for about 22% of the energy consumed in buildings. The remaining is connected to the building construction industry (17%) [2].

Methods for energy prediction, energy models, have in general two purposes in a building: design or optimization (of for instance the HVAC system) before and during construction, and calculating savings for retrofitting strategies or enabling model predictive control in existing buildings [2]. Demand forecasting and decomposition of energy consumption patterns can help identify the major objectives for energy conservation. Furthermore, sufficient energy predictions can help building managers shift energy consumption to off-peak periods, make more efficient energy purchase plans, form energy storage, and utilize energy buffers, that in sum can help reduce the total energy consumption, increase the usage of energy from renewable sources, and reduce the costs of energy [3]. Models for energy prediction could also help detect abnormalities inside the building, which in turn can reduce the response time for managing the cause.

The time people spend in an indoor space is today about 90%, making the indoor environmental quality (IEQ) an important parameter of the quality of life for the occupants [4]. Energy conservation, which can lead to good energy efficiency without sacrificing comfort levels, does then particularly mean identifying areas of wasteful energy usage and then taking actions to reduce this surplus [1].

Models for energy predictions can serve different purposes depending on the area of need, and not only on a building specific level. Governments or city planners can for instance also benefit from using energy models for high level predictions like the future energy demand in cities or different districts, power grid requirement and free capacity simulations, and much more.

Globally, heating is a larger energy consumer compared to cooling today, but some research would suggest that global warming and climate changes will transition the energy usage from heating to more cooling in the future, making both the pattern and geographical location for consumption change over time.

1.2 Energy and Thermal Models

There has been an increasing interest in developing models for energy consumption and thermal behavior for buildings for the last decades. These models can usually be divided into three main categories: white, grey, and black box models (according to [5] and [2]).

1.2.1 White box model

White box models are based on mechanistic models that require calibration of the physical parameters based on building specific material knowledge. One key advantage for this type is that models can be created without the use of recorded building data, which again uncover one of the challenges of estimating correct parameters for the model. The degree of complexity can vary in these models with one strategy being lumping parameters together. There exist several computer programs that utilize these kinds of models for energy simulation.

1.2.2 Black box model

Black box models represent the data-driven approach to modelling, whereas different recorded features are used as inputs to the model and the model is calibrated to predict the correct output. The black box models are therefore not reliant on a physical understanding or differential equations, but there is often not a straightforward way of understanding how the predictions work, hence the name “black box”. A machine learning model is a data-driven model type and will fall under this category.

Lately, there has been a development in this field with algorithms that can help understand how these machine learning algorithm’s function. For instance, what features a neural network put emphasis on during image recognition. Which could make the workings of such algorithms a bit less of a “black box”.

1.2.3 Grey box model

Grey box models can be seen as sort of a combination of the white and black box approach where the models are typically mechanistic models, of the same structure as one can find in white box models, but the identification of model parameters are found using data.

One popular example of a grey box model is the lumped capacitance model, where the distributed thermal mass of the building is “lumped” into a discrete number of capacitances. Inside the model, thermal resistors interconnect these thermal capacitors, making it a thermal network. Therefore lumped capacitance is often referred to as “thermal networks”, “resistor-capacitor” or simply “RC networks”.

1.2.4 Important parameters

The total energy consumption in a building will depend on the equipment responsible for consuming energy, this can be any physical equipment, machinery, process, or a combination of these [1]. Figure 1-1 illustrates how some of the different energy consumers and other parameters can affect the total energy consumption of a building.



Figure 1-1: Important parameters in energy consumption

A study from 2012 shows that the energy consumption in buildings can be divided into categories as shown in Figure 1-2 [1]. In this study, lighting is shown to be a substantial factor. It is reasonable to assume that this percentage would be reduced today due to the increased use of low energy lighting such as LEDs.

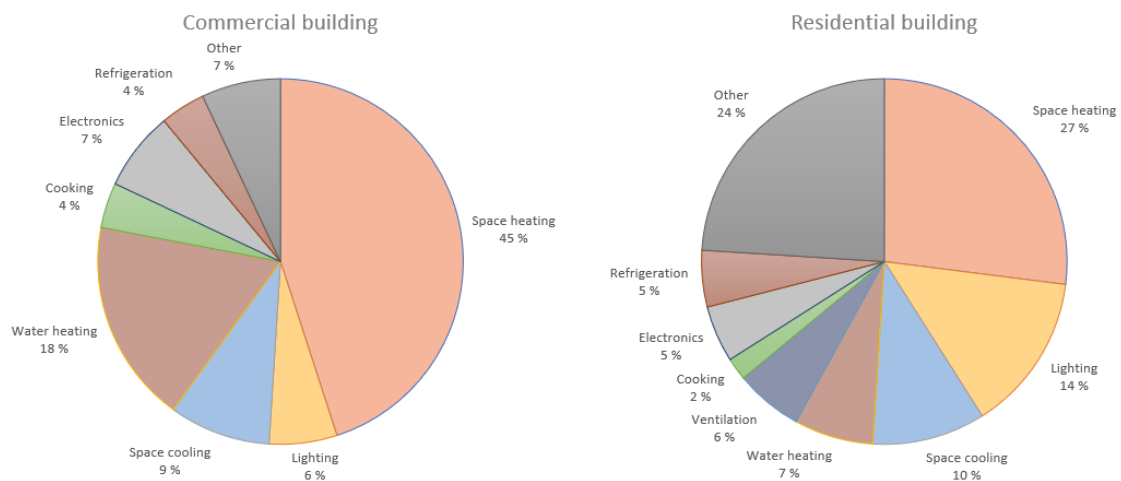


Figure 1-2: Energy consumption in buildings [1].

How these percentage of energy consumption is divided for a specific building will differ depending on the usage of the building, geographical location and so on. Some of the general important parameters that influence energy consumption is [1]:

- Climate related
 - Solar radiation
 - Air temperature
 - Wind characteristics
 - Sky or cloud conditions

- Building location
- Building or usage
 - Room air temperature
 - Thermos-physical properties of construction elements
 - Internal heat gains
 - Ventilation rate

A report from Enova (governmental energy advisor organization in Norway) states that the percentage of energy consumption in Norwegian buildings due to outdoor temperatures will greatly depend on the building type (residential, office, school...) but even more important is the building standard used during construction. As an example, for an office building built in the 1950's as much as 55% of the energy consumption is due to outdoor temperature conditions, compared with only 20% for a building built in the late 2010's [6]. This shows that the particular variable importance, with respect to energy consumption in buildings, will probably be building specific. This report from Enova refers to Norwegian building standards and climate conditions where cold temperatures and heating of buildings are important, but it is assumed that the same can also be true for a warmer climate where cooling is more relevant.

1.2.5 Related work

There exists a lot of research in the field, and many have earlier successfully created models within the gray box domain. Since the grey box models are only partly data-driven, recent reports then often compare the use of grey box models to complete data-driven models (black box models). The model types that are commonly used for the black box approach are then ANN, SVM and ARX type, like in [7] and [2]. Here, they concluded with the ANN model type giving the best predictions.

1.3 Project objectives

The project's objective is to identify important parameters related to energy consumption in buildings and make machine learning models that can predict the future energy consumption in a building. Compare the result from different machine learning modelling techniques, and also compare it to a simpler modelling technique. This project will focus on making energy prediction models for commercial buildings. The full task description for this master thesis can be found in Appendix A.

1.4 Report structure

This report follows the IMRaD (Introduction, Method, Results and Discussion) format. The methods chapter begins with a theory overview of machine learning, with emphasis on the methods used during this project. Then describes how the raw data is managed, prepared, and further analyzed, before lastly describing the models and implementation of models. The results chapters will present the data analysis and modelling results, and finally the discussion and conclusion chapter will discuss and comment on the overall findings during the project.

2 Methodology

This chapter starts with a description of some of the background theory for machine learning, with a particular focus on the modelling techniques used in this project, and how machine learning relates to artificial intelligence. Additionally, some information about implementation and the main libraries used during code implementation. The chapter continues by describing the specific methodology used during the project work and model development. The overall project development can be broken down into three steps, first importing and pre-processing of raw data, secondly data analysis to transform data into information by identification of key features within the data and features engineering, lastly the development of different machine learning models based on the features identified during data analysis.

2.1 Artificial Intelligence and Machine Learning

This subchapter will introduce Artificial Intelligence (AI) and Machine Learning (ML) in general, how they relate, and some of the terminology used in this context. In addition, some of the theory behind the specific ML methods used in this project.

First of all, what is artificial intelligence? Artificial intelligence can be described as “*The effort to automate intellectual tasks that are normally performed by humans*” [8]. This means that AI includes a broad field, which may not include any learning at all. Consider the early chess programs in the 1980’s, these were strictly rules-based reasoning programs hardcoded by skilled programmers. This approach is now known as symbolic AI.

Machine learning is a subfield of AI, so you could say that machine learning is AI. Machine learning is different compared to symbolic AI, as this method is not dependent on human programmers to write down specific rules – a computer program, in order to turn inputs into suitable outputs or responses. Machine learning rather turns this process on its head and turns inputs and outputs into rules, so ML is a process of *training* rather than programming. In a way ML is just mathematics which can especially relate to statistical mathematics. But unlike classical statistical analysis, ML tends to manage very large datasets and is a field mostly driven by empirical findings which is greatly reliant on improvements in hardware and software [8].

Training a machine learning algorithm is usually performed by repeated exposure to a series of examples. Where the ML algorithm is given one or several inputs and the model’s current output is compared to the expected output (known output) to see how well the algorithm is performing. The result from this comparison is then used in a feedback loop to adjust and enhance the algorithm’s predictions through a process referred to as *learning*. So, through repeated exposure to known input and outputs the machine learning algorithm *learns* to transform inputs into meaningful outputs. The core of machine learning is therefore this process of transforming inputs into meaningful representations of the data that gets us closer to the expected output [8]. The finished product of machine learning is then a *model*, a model found (or *learned*) out of data through a process of training [9].

This brings us to one of the core challenges with machine learning, the *training data*. No machine learning method can produce a good model or desired goal without adequate, unbiased data that accurately reflect the process which it is to model. The process of getting the model to produce consistent predictions on all data, is called *generalization* [9]. In addition to using

2 Methodology

incorrect training data, another problem that can disturb the model’s generalization capabilities is called *overfitting*. Overfitting is when the model is overly adapted to the training data, meaning that in addition to learning the process dynamics also learn noise and outliers in the training data and hence lose some of the generalization capabilities. This would often result in good prediction results on training data but poor predictions for general data [9]. The opposite of overfitting is *underfitting*. Underfitting can take place when the model is not adequately trained and has not yet sufficiently “learned” all the dynamics of the process. This would normally give poor predictions results on both the training and general data.

In the world of machine learning, there are two basic approaches: *supervised* and *unsupervised learning*. Supervised learning applies labeled data to learn “known” outcomes, whereas unsupervised does not use pre-labeled data but rather finds patterns within the data [10]. There also exists semi-supervised which is a combination of the two, and reinforced learning [9].

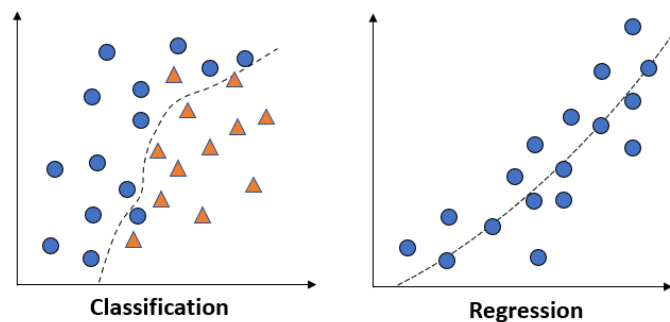


Figure 2-1: Classification and regression problem principal [9].

One can typically divide supervised machine learning problems into two main tasks, classification, and regression problems. In a classification problem, the task is to divide data into compartments (or classes) and then assign the correct class label [11]. Examples of classification problems can be image classification where the goal is to classify if an image contains a “cat” or “dog”, identify if an email is “spam” or “not spam”, or it can be to classify if a person can be “trusted” or “not trusted”. Whereas the classification problem predicts discrete class labels, the task in a regression problem is to predict a continuous value based on the input variables. Figure 2-1 illustrates the difference between the two problem types.

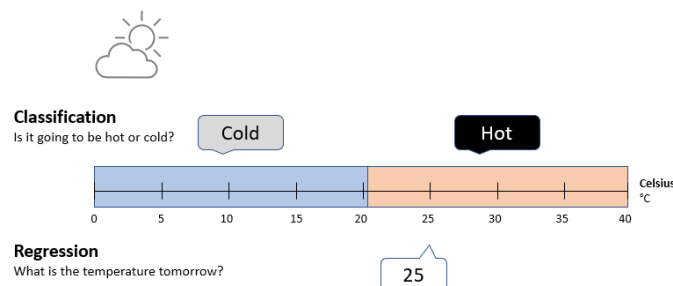


Figure 2-2: Comparison of classification and regression task [11].

An example of the difference between classification and regression is given in Figure 2-2, where the classification part is to classify temperatures into either “hot” or “cold”, whereas the regression problem is to determine the temperature as a numerical value.

Inside the domain of machine learning, there are many different techniques. While many of these techniques are fairly old, the field of machine learning, and especially deep learning, has

2 Methodology

only gained attention in the last decade due to the many remarkable achievements particularly in the fields of speech and image recognition. As machine learning is guided by experimental findings rather than solved theoretically with a pen and paper, it can be viewed as more as an “engineering science”. Most of these recent achievements are due to increasing processing powers (hardware), datasets availability and the boost in investments in the field [8]. Figure 2-3 shows how AI, ML and some of the ML techniques relate.

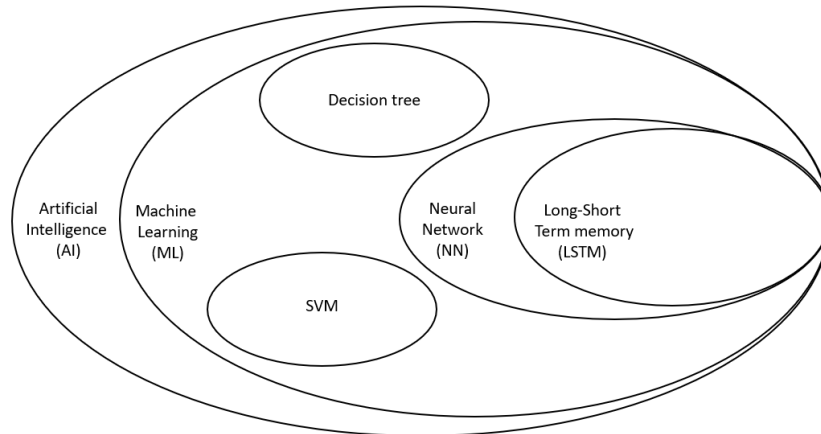


Figure 2-3: Relationship of artificial intelligence, machine learning and some ML techniques.

2.1.1 Timeseries and Machine Learning

The term *timeseries* is used for a sequence of sample values indexed in a timely order, often with an equal time interval between making it a sequence of discrete values. A timeseries could thus be any data obtained through measurements on regular time intervals, like the outside temperature, price of a stock, or energy meter value. The most common task when it comes to timeseries and machine learning, is *forecasting* – predicting what’s happening next in the timeseries [12].

Timeseries forecasting is a bit different compared to other machine learning tasks as it is important to know the dynamics of the system, understand the trend, seasonal, cyclical, and random variations in the system. It is also important to identify some patterns or features that are related to the outcome in order to make any successful forecasting, and it would for instance be impossible to make a good forecast on a random signal.

2.1.2 Training a Machine Learning algorithm

Before training begins, the complete dataset is normally divided into *training*, *validation*, and *test sets*. The training set is used for training the model, the validation set is used for tracking the training progress, and the test set is only used after training to test the model or compare to different models before production. When working with timeseries forecasting it is also important that the validation and test set is more recent than the training set, as we try to predict the future given the past and not the other way around.

It is normal to use the same training data several times during one training session as the learning method is iterative and gradually improves the model’s performance. The term *epoch* is used to describe how many iterations the complete training set has been shown to the model,

where one *epoch* refers to one iteration. For a training session, the number of epochs is normally specified in advance. *Early stopping* is referred to as when the training is stopped before reaching this specified maximum number of epochs (hence the name “early stopping”) due to deficiency in learning rate for the validation set. Early stopping is used as a tool to prevent overfitting. Figure 2-4 illustrates how the number of epochs will typically impact the learning and prediction error for the training and validation set.

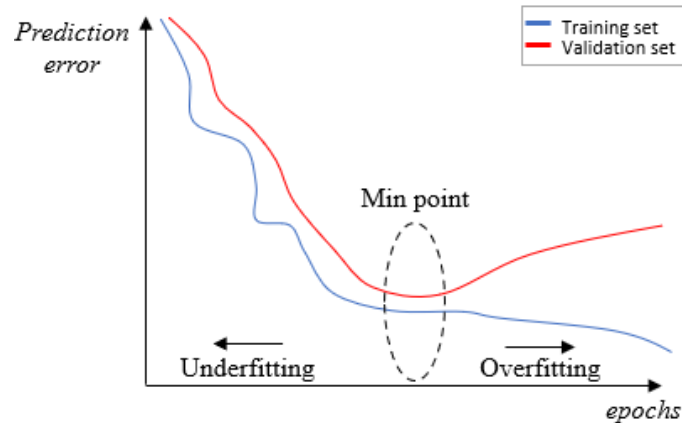


Figure 2-4: How number of epochs typically affect ML training.

The tunable parameters that define the model structure and those used to control the learning process are called *hyperparameters*. There is no single perfect set of hyperparameters that will work for every model, so identifying these hyperparameters can be the real work of tuning the ML model.

2.2 Neural Network - NN

Neural network is a type of machine learning algorithm. The structure of a neural network (NN), or more precisely an artificial neural network (ANN) in this case, is inspired by the working principle of the human brain where many biological neurons are interconnected in a large network to process inputs and extract information [9]. The artificial neural network is, like the name suggests, a network of neurons with one of the core components being the artificial neuron or *node* as they are called with regards to ANN.

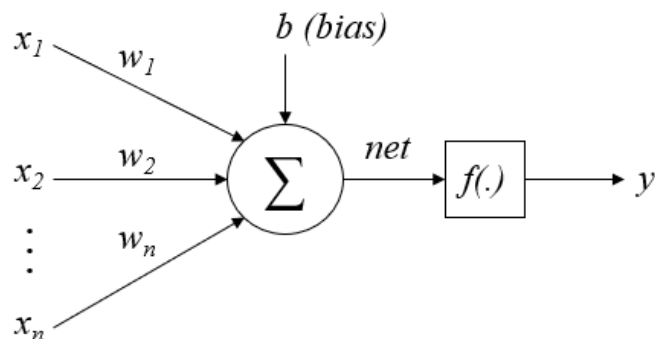


Figure 2-5: Working principle of a node in an neural network [12].

2 Methodology

A node is actually a mathematical function where it takes inputs (x), weights them individually (w), sums them up and adds a bias factor (b) to calculate a net sum. This net sum (net) is then passed through an activation function ($f(.)$) to produce the node's output (y). The calculation of the net value is shown in equation (2.1) and (2.2). See Figure 2-5 for illustration of node working principle.

$$net = \sum_{i=1}^n (w_i \cdot x_i) + b \quad (2.1)$$

Can also be written as:
$$v = w \cdot x + b \quad (2.2)$$

Where:

$$w = [w_1 \quad \cdots \quad w_n]$$

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

There are several types of activation functions like the linear, step, ramp, tan-sigmoid, sigmoid, ReLU [12], examples shown in Figure 2-6.

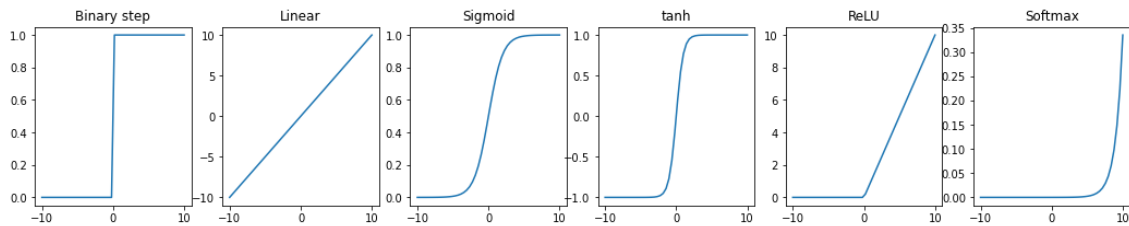


Figure 2-6: Examples of different activation functions.

Nodes are usually organized in *layers* where nodes are connected to form a neural network. In this layered structure, information enters at the input layer, passes through the hidden layers, and exits at the output layer. During this process the information is processed layer by layer, where all nodes in a layer receive information, process it, and create the output simultaneously [9], see illustration in Figure 2-7.

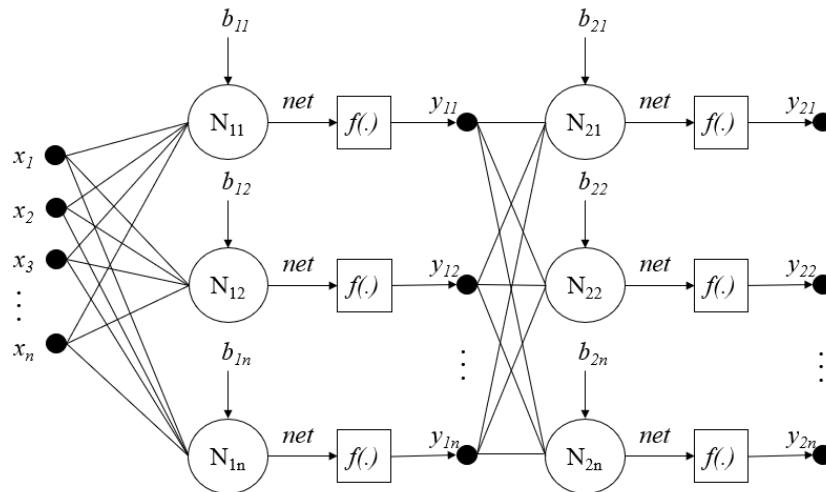


Figure 2-7: Multi-layer perceptron network [12].

The structure, the number of nodes in each layer, and the way they are connected will depend on the task at hand. Networks with a single hidden layer are called *vanilla neural networks* or *shallow networks*. When more hidden layers are added (at least two hidden layers), it becomes a *deep neural network*. If all nodes in the previous layer are connected to every node in the next, we call it a *fully connected network*. This fully connected *feed forward* architecture is the standard architecture and most used for simple applications, where feed forward networks only consider forward connectivity [13]. See Figure 2-8 for an illustration of the general structure.

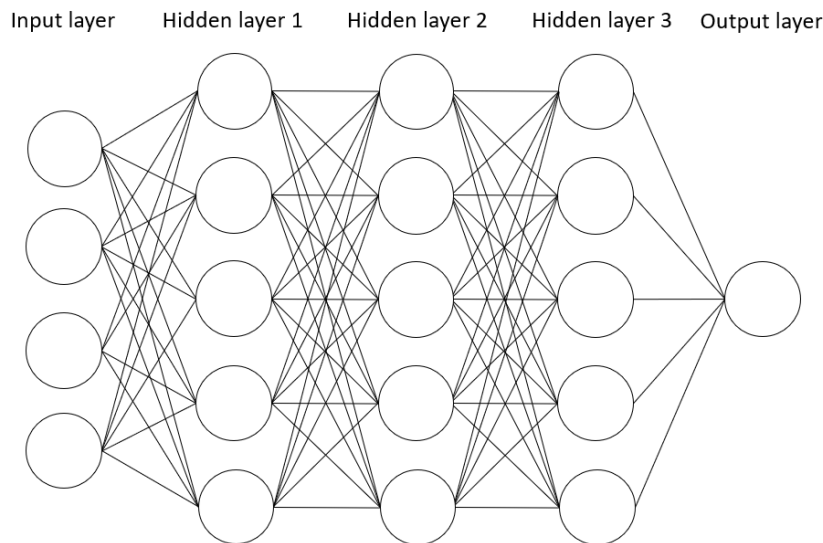


Figure 2-8: General structure of multi-layer fully connected neural network.

Adding more nodes or layers can increase the network complexity capabilities, but also increase the computational load of training in addition to the risk of overfitting.

2.2.1 Training the neural network

During neural network training a *loss function*, sometimes also called *objective function* or *cost function*, takes the network's output, compare it to the true output and computes a distance

2 Methodology

score [8]. This score is then used in a feedback loop to adjust the network's weights in a way that tries to minimize this score. Two of the main types of such cost functions are sum of squared error and cross entropy function [9].

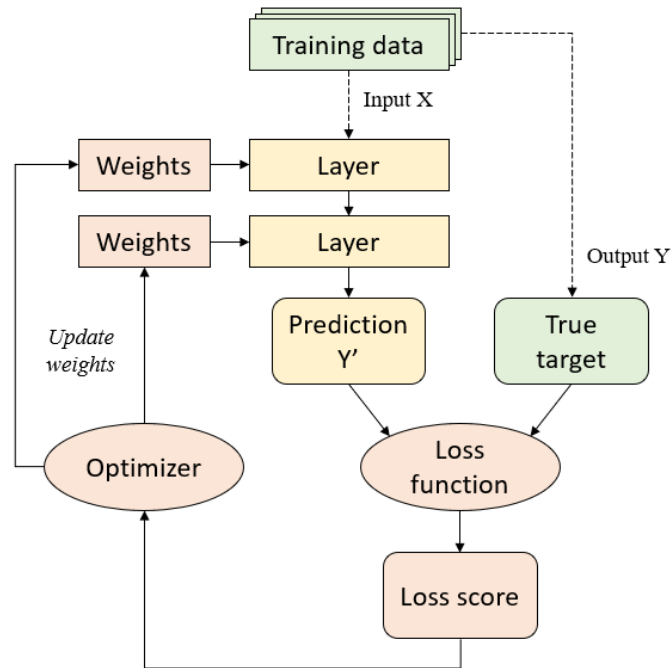


Figure 2-9: Flow chart of the NN training process [8].

This process of adjusting the weights and biases based on the loss score is the job of the *optimizer*, which applies what is called the backpropagation algorithm. There exist several different optimizer algorithms, such as: Stochastic Gradient Descent (SGD), batch, mini-batch, Adagrad, RMSProp, AdaMax, Adam. Which optimizer to use depends on the task as all have their strengths and weaknesses [8]. Figure 2-9 shows a flow chart of a neural network training process. The optimizers will adjust the weights and biases so that the loss score is minimized, and hence the prediction error of the model will improve. One tool the optimizers can use is to follow the gradient descent (imagine a ball rolling downhill as shown in Figure 2-10). One potential problem is then to get stuck in local minimum points. To help solve this local minimum problem the concepts of *momentum* and *velocity* are utilized by some optimizers, as where the ball has large momentum, it can help roll over the local minimums and continue to reach the global minimum point.

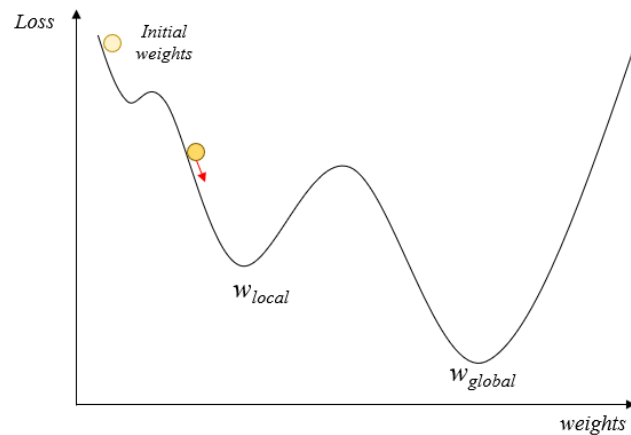


Figure 2-10: Concept of gradient descent

Learning rate is a measure of how quickly the model should learn during training, or more precisely how much the weights should be adjusted each time. A too large learning rate will adjust the weights too aggressively and will in many cases struggle with finding the actual minimum point. Therefore, it can be sensible with a too small learning rate rather than too large, but it varies. Figure 2-11 shows an illustration of how different learning rates can affect the learning process. There can be a tradeoff between the computational load and training time as a lower learning rate must be compensated for with a higher number of epochs.

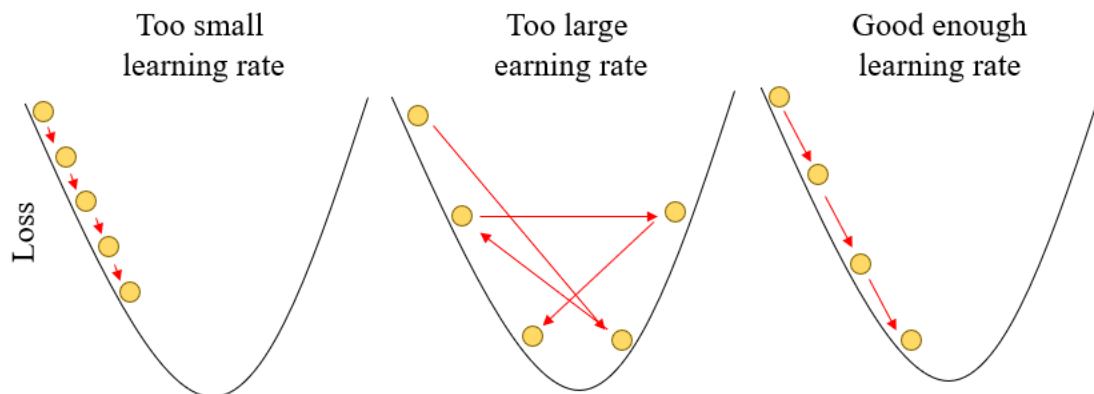


Figure 2-11: Illustration of learning rate.

As earlier mentioned for ML in general, the tunable parameters that define the model structure and those used to control the learning process are called *hyperparameters*. With regards to neural networks, the number of layers, number of nodes in each layer, optimizer and learning rate are all examples of hyperparameters.

2.2.2 Recurrent Neural Network (RNN)

Recurrent Neural Network (RNN) is a type of neural network with capabilities to “remember” past *states* or inputs, which makes it good when processing sequential data like in timeseries forecasting, speech recognition, or other tasks where earlier inputs will influence the future. In contrast to the traditional feed forward architecture, where incoming data travels in a single direction from input to output, the RNN structure includes feedback loops to preserve previous information [14]. Consider a piece of a RNN, A , which has some input x and some output h as

shown in Figure 2-12. By including the feedback loop, this allows for information to be passed from one step to another.

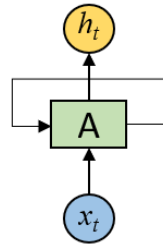


Figure 2-12: Recurrent neural network, feedback loop

This loop structure makes it possible to think of the recurrent neural network like the normal neural network structure if we “unroll” the loop in time [15], see Figure 2-13. Since the RNN has a loop structure, all the modules in the chain share the same parameters across each layer [16].

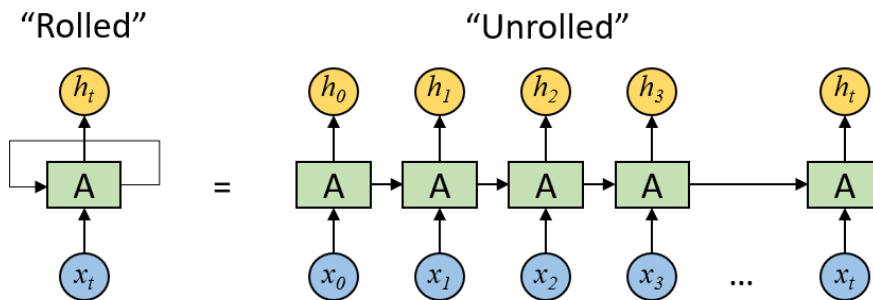


Figure 2-13: An unrolled recurrent neural network [15].

If we unroll the RNN it takes forms as a chain of repeated module structures. In the general RNN structure as discussed here, these repeated modules will usually be quite simple and contain a single tanh layer, see illustration in Figure 2-14.

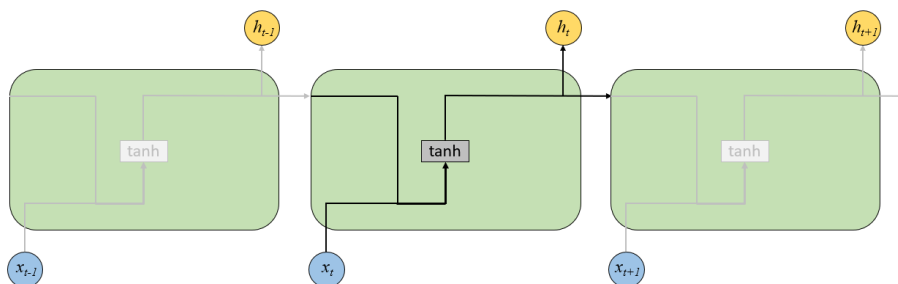


Figure 2-14: Repeated module structure in chain for general RNN [15].

A frequent problem when training RNN is the *exploding* or *vanishing gradient*. When the gradient is too small, it continues to get smaller, and updating the weights will finally set it to an insignificant value which effectively makes the network stop learning. Exploding gradient is the opposite, when the gradient is too large, the network becomes unstable. One solution to vanishing and exploding gradient, can be reducing the number of hidden layers in the model and thus reducing the complexity [17].

The appealing idea of the RNN is that it is in theory able to link previous information to the present task, but in practice this is only partly true as it can be useful if the time frame is narrow, meaning the information is not too far into the past, unfortunately many real life situations require a longer “memory”. This is where another special type of RNN comes in, that is the Long Short Term Memory network (LSTM) [15].

2.2.3 Long Short-term memory (LSTM)

As introduced in the previous subchapter, LSTM is a special type of RNN but in addition to the short term memory capabilities of the general RNN, the LSTM also has the ability for a longer “memory”, hence the name long short term memory. The structure of the LSTM is like the general RNN structure, a chain of repeated modules, only with a different module content. In comparison to the ordinary RNN, LSTM does not struggle as much with vanishing or exploding gradient. The general structure of the LSTM is shown in Figure 2-15.

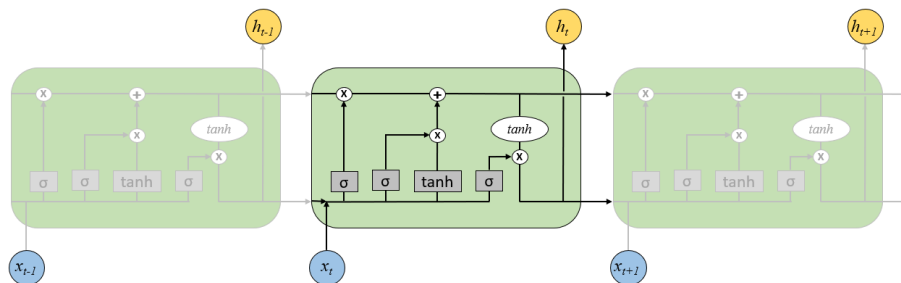


Figure 2-15: Repeated module structure of chain in LSTM [15].

The LSTM block contains four neural networks, in combination these networks have the ability to select what information to add or remove from the *cell state*. This operation is managed by three *gates* which in turn controls how much information should be forgotten (*forget gate*), added (*input gate*) and outputted (*output gate*) from the *cell state*. Where the cell state c_t is the previously mentioned “memory” of the LSTM network. Gates are put together out of a sigmoid neural network and a pointwise multiplier as shown in Figure 2-16. The sigmoid network layer will produce a value between zero and one, describing how much information to let through the gate. Also see detailed illustration of LSTM block in Figure 2-17.



Figure 2-16: Structure of a LSTM gate

The decision of how much of the cell state information should be kept or forgotten is based on the previous output h_{t-1} and current input x_t and is controlled by a sigmoid neural network called the forget gate layer. This sigmoid network produces a value between zero and one and will output one in order of keeping all cell memory and zero for forgetting cell memory.

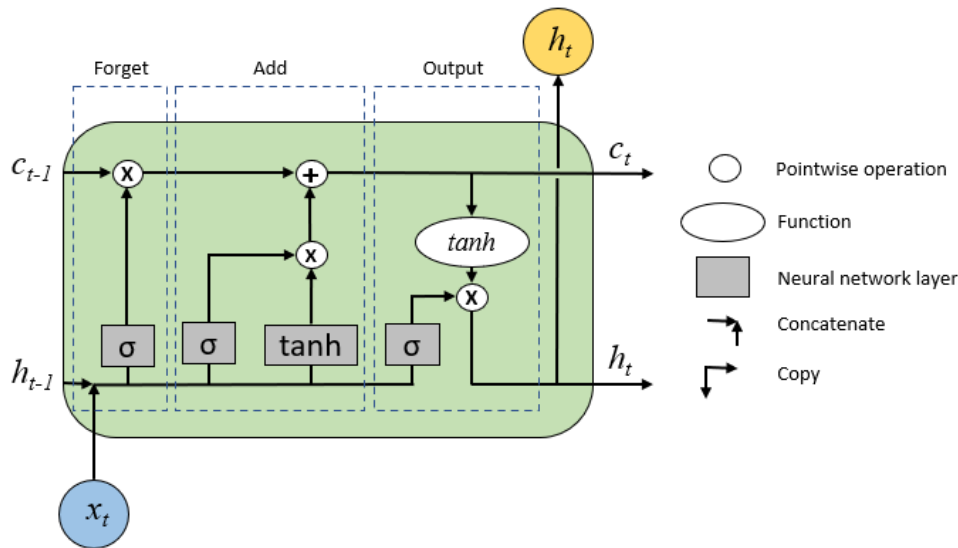


Figure 2-17: Detailed view of traditional LSTM block.

To control how much new information should be stored in the cell state, a combination of a sigmoid and a tanh neural network is used. The tanh network calculates potential candidate values to be added to the cell state from a combination of current input x_t and previous output h_{t-1} . Then the sigmoid net decides which of these candidate values calculated by the tanh net to be added to the cell state.

Lastly the output is controlled by a multiplication of a tanh function and a sigmoid neural network and is a “filtered” version of the cell state. Where the sigmoid network decides what parts of the cell state to output based on the previous output h_{t-1} and current input x_t . The tanh function takes the current cell state and compresses it to a value between -1 and 1.

There also exist other variations of the LSTM cell, one of which is the Gradient Recurrent Unit (GRU), which is found to cost less computation and give comparable results. For more information about the LSTM network, variants, and the math behind, see [15].

2.2.4 Deep learning

Deep learning is a subfield of machine learning that usually employs deep neural networks (as earlier mentioned, neural networks with two or more hidden layers) [9]. Lately deep learning has become the hero of machine learning due to the many advancements, such as in image and speech recognition.

Deep learning will typically have a higher probability for overfitting due to increased network complexity, but this tendency can be mitigated by using massive amounts of training data in addition to using *dropout nodes* while training and regularization.

One potential showstopper for deep learning is the dramatic increase in computational load as the networks and the data amounts become larger. The use of GPUs and even TPUs for training can help reduce training times, but the elephant in the room is still the energy consumption related to the massive computational load both for training and running applications like the lately blooming ChatGPT [19].

2.3 Decision Tree - DT

A Decision Tree (DT) is another type of machine learning algorithm, where the structure is like a flowchart with a series of hierarchical questions. The decision tree begins with a *root node* (without any incoming branches) and depending on the choices follows through *branches* and *leaf*, in total organized as a tree. From the root node the branches go to *internal nodes*, also called *decision nodes*, before ending up in *leaf nodes* or *terminal nodes*. The internal nodes represent a test on an attribute, and the leaf nodes represent all the possible outcomes or predictions of the decision tree. The general DT structure is illustrated with a classification problem in Figure 2-18.

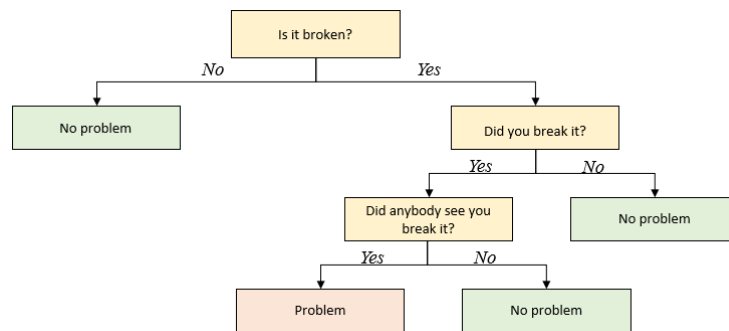


Figure 2-18: Example of decision tree structure, like a flow chart.

This simple structure makes the decision tree easy to interpret and visualize. Decision trees can be used both for classification and regression tasks. The process of training the decision tree is the process of identifying the ideal splitting points of the tree. This process is repeated until the majority of the training data is correctly classified. As the tree grows (by adding branches and nodes), data can be fragmented into too many sub trees which can easily make DT be overfitted. Therefore, to preserve its purity, decision trees should be kept as small as possible [18].

2.3.1 Random forest

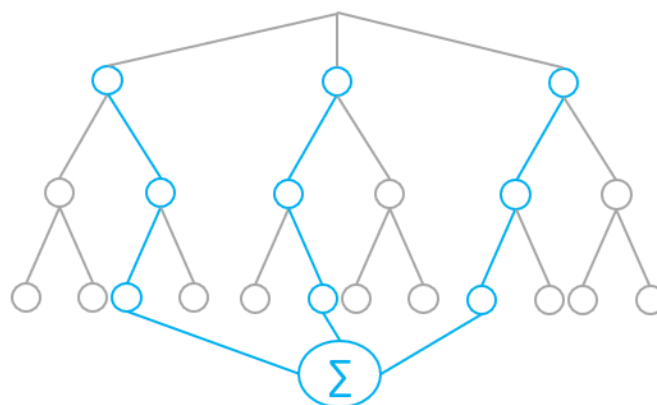


Figure 2-19: Working principle of random forest.

Random forest is an assembly of multiple decision trees used to predict one single output. This ML technique is used to improve the prediction accuracy of single DT, this technique is especially effective when the individual trees are uncorrelated. One tool used to ensure that the trees stay uncorrelated, is to only use a selection of the features (not all) when training each

individual tree, or only use a selection of the training data for each tree [19]. The working concept of a random forest is illustrated in Figure 2-19.

2.3.2 Boosting

Boosting is the process of building one strong classifier based on a combination of many weak learners. Boosting is a method that can be used for improving any given machine learning algorithm. When boosting decision trees, the trees are trained in a sequential order in contrast to *bagging* where trees are trained in parallel (as for random forests training) [20].

Adaptive boosting (AdaBoost) is a technique that applies boosting [21]. In contrast to random forest which consists of several full-size trees, adaptive boosting usually just employs nodes with two leaves (also known as a *stump*). The adaptive boosting technique consists of:

1. Start by making a simple model based on all the training data.
2. Then the misplaced samples from this initial model are given extra weight (“boosted”) and used to train a new simple model
3. Train a new model with the previously adaptively boosted samples. Then the misplaced samples from this model are given extra weight and used to train a new simple model. Each new tree will then focus on the prediction errors from the previous.
4. Step 3 is repeated until the maximum number of trees is reached, or all samples are correctly classified.
5. Use output from all trees for total prediction.

2.3.3 Gradient boosting machines

Gradient boosting is similar to the previously described adaptive boosting, but each tree is typically larger. One key feature is that the technique focuses on the residuals from the previous prediction and are therefore called a gradual learner. The gradient boosting regression tree consists of [22]:

1. The first prediction (base prediction) is the mean target value for all training data.
2. The second prediction will then focus on the residual error between the mean value (previous prediction) and the actual values.
3. The next layer will then focus on the residuals for the previous layer and predict this.
4. Step 3 continues to add more simple trees until the maximum number of estimators are reached, or the learning stops improving.
5. Use output from all trees for total prediction.
 - a. The models are also weighted to adjust how fast each model should contribute through the learning rate parameter.

2.4 Python

Kaggle is a website that, among many things, hosts popular competitions for machine learning algorithms worldwide. In addition to hosting competitions, Kaggle also runs a yearly survey among its machine learning community to identify which algorithms and libraries they use. The most popular libraries or frameworks in 2020 were Scikit-learn, TensorFlow, Keras and XGBoost. One thing all of these have in common is that they are all Python libraries. Today

Python is by far the most used language when it comes to data science and machine learning, possibly because of the vast number of libraries to use [8]. This subchapter will introduce some of the Python libraries utilized throughout this project.

2.4.1 Google Colaboratory – Colab

First of all, when coding you need a code editor. The Python editor used in this project is Google Colaboratory – Colab. The programming environment in Colab is like a Jupyter notebook where it is possible to combine text and script in blocks. Using Colab has several benefits, working documents can automatically be stored in the cloud, code can run on a server and not on the local machine meaning libraries are installed on the remote server and with potentially larger processing powers (can also utilize GPU and even TPU for training machine learning algorithms).

2.4.2 TensorFlow and Keras

TensorFlow is an open source, Python based, platform for developing machine learning projects, where its primary contributor is Google. The platform is designed to be a convenient way for engineers, researchers, and others to utilize neural networks and especially deep learning. The projects can run on normal CPUs, but also on GPUs and even TPUs (highly parallel hardware accelerators). TensorFlow programs can also be exported to run on other runtimes C++ and JavaScript (web browser applications) or by using TensorFlow Lite run on mobile or embedded devices [8].

Keras is an API built on top of TensorFlow. Keras was originally made for research purposes for easy and fast experimentation with deep learning. Keras is made easily understandable for humans (not only machines), and therefore based on simple and easy-to-understand commands. The Keras library has over a million users today with many of the largest companies in the world on its users list [8]. In this project the Keras library is used to create and train various neural network models. Some of the important parameters of the network being:

- **Layers:** Structure of layers, neurons in each layer, activation function
- **Optimizer:** Adam...
- **Learning rate:** example 0.01
- **Loss function:** example: MAE

2.4.3 XGBoost

XGBoost (Extreme Gradient Boosting) is a popular library for implementing algorithms in the gradient boosting framework. The library is optimized to be efficient, portable, and flexible [23]. In this project the model “*XGBRegressor*” is used, with some of the most important parameters being [24]:

- **n_estimators:** The number of trees in the assembly
 - normally increased until the predictions stopped improving.
- **max_dept:** The maximum depth of each tree
 - Usually, a value between 1 and 10.
- **objective:** The objective for the algorithm

- For regression problems typically “*reg: linear*”
- **learning-rate:** The weight assigned to each model (tree) added. This means we can add more trees before the model overfits. In general, it is better to have more estimators with less weight on each.
 - Usually set to a small number 0.1, 0.01... (1 is maximum).
- **subsample:** The number of samples used to train each tree.
 - Value between 0-1.
- **colsample_bytree:** Number of features used in each tree.
 - Value between 0-1, 1 being all features.
- **early_stopping_rounds:** Specify number of rounds with no improvements on validation set before stopping training.
 - Value of 5-10 is a reasonable number.

2.4.4 Scikit-learn.

Scikit-learn is an open-source ML library that supports multiple supervised and un-supervised learning algorithms, it also includes many useful tools used during machine learning projects. In this project mostly the supporting tools in this package are used, not the models.

2.4.5 Pandas

Pandas is an open-source library in Python, it is not directly a machine learning library, but it is typically used for preparation of data and data analysis in relation to ML. Some of the highlights are efficient tools for reading and writing to and from files of various formats (CSV, JSON...and more). The data can be handled in data frames, where data is displayed as a two-dimensional table. The library contains tools such as indexing, sorting, filtering, splitting, merging of data frames, in addition to statistical calculations such as min, max, mean, standard deviation to name a few. This makes it a widely used library within many domains of handling data [25].

2.4.6 Matplotlib

Matplotlib is one of the most popular libraries when it comes to data visualization. Much like Pandas, Matplotlib is not directly a machine learning library, but it is extensively used since it provides a large diversity of visualizations. The Seaborn library is also used as a supplement to Matplotlib for some of the visualization.

2.5 Data preparation

This subchapter will present the methods used for handling raw data and the pre-processing performed before the data can be further analyzed and used in models.

In this project, energy data recorded from two buildings located in Athens, Greece in the period from 2020-01-01 to 2022-12-31 (UTC) is used alongside weather data for Athens covering the same time period. The provided weather data for the project has the weather service “Open Weather Map” as its origin.

Both buildings consist of several energy meters measuring the electrical energy consumption from various energy consumers inside the buildings, where the parameter “active energy” is the parameter used from these energy meters. This parameter is given as the total energy consumption (in kWh), which is an incremental value, normally over the whole energy meter’s lifetime. Table 2-1 and Table 2-2 shows an overview of energy meters in the two buildings. Except for the fact they are both commercial buildings, there is no additional information available about the buildings like: buildings usage type, occupancy level, building material, floor layout. The logical naming of each energy sensor is performed by the building owners and can be arbitrary. Figure 2-20 illustrates how energy meters and different energy consumers could typically be distributed within an example building. The total energy consumption for a building is then the combined energy consumption for all energy meters within that building - the sum of all energy meters.

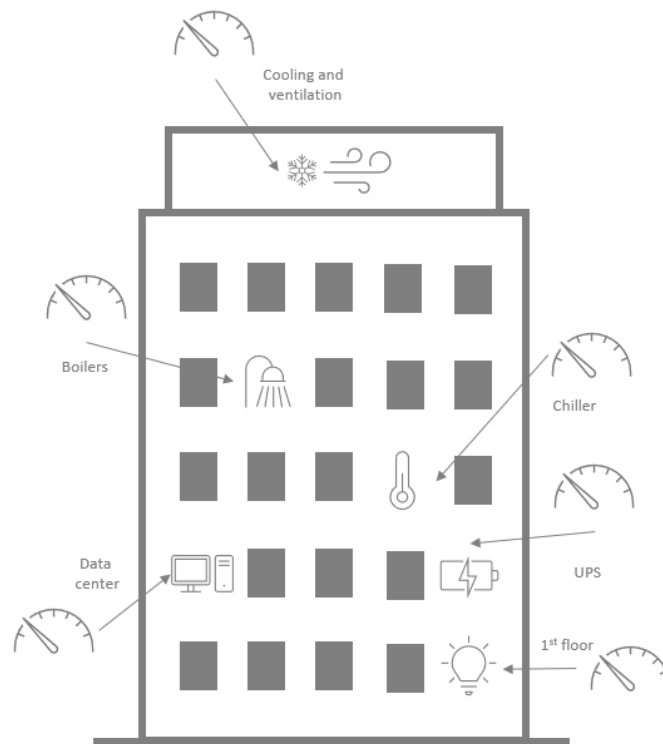


Figure 2-20: Illustration of energy meter distribution in an example building.

2 Methodology

Table 2-1: Energy meter overview, building #1

Logical Name	Format (columns x rows)
1 st floor	2x100751
Air-condition ventilation #Q7E	2x100747
UPS #02	2x100719
UPS #01	2x100694
Boilers #Q1.2	2x100682
Field Switch Generator #01	2x100675
Field Switch Generator #02	2x100661
Chiller #01 Q2.1 Calliroe Street	2x100646
Chiller #02 Q2.1 Svvgrou Street	2x100634

Table 2-2: Energy meter overview, building #2

Logical Name	Format (columns x rows)
P2 Power Meter	2x285908
PUCR2 Power Meter	2x286813
PUCR New Power Meter	2x509450
P1 Power Meter	2x286925
PUCR3 Power Meter	2x286867
PUCR1 Power Meter	2x269847
PUCR Power Meter (16)	2x287191
Chillers Data Center Power Meter (17)	2x284424
Triple Switch Power Meter (18)	2x510069
Chiller 1 Data Center Power Meter (21)	2x510090
Power Transformer 2 Power Meter (19)	2x505201
Power Transformer 1 Power Meter (20)	2x509581
Chiller_1 Power Meter (23)	2x734647
Chiller _2 Power Meter (22)	2x761791
Building Chillers Power Meter (24)	2x509588

2.5.1 Raw data format

The raw data provided is in a number of CSV- and JSON-files, and all of them must be read into Python for further processing. The energy data consists of several files, one file for each individual energy meter in a building. The general format of these energy data files can be seen in Table 2-3.

The energy data files:

- Building #1: includes data from 01.01.2020 to 31.12.2022. About one record per 15min.
- Building #2: includes data from 10.01.2020 to 31.12.2022. Most files have about one record every 5 minutes, some files with one record every other minute.

Table 2-3: General format of all energy data files

Feature name	Unit	Sample format	Description
kWh	kWh	2862855.75	Energy meter value, incremental value of total energy consumption
Time	Datetime.	2020-01-01T00:04:59Z	Timestamp for meter reading, in UTC time according to ISO 8601 ¹ formatting.

The raw weather data has a format of 20 x 29 341 columns and rows originally (empty columns are dropped during import). The data resolution is approximately one record per hour, but the time shifts during the dataset, meaning it is not the same minute for each hourly record hence not exactly one hour apart. The general format of the weather data can be seen in Table 2-4.

¹ ISO 8601 is an international standard covering exchange and communication of date and time-related data. The Z at the end of the time stamp indicates that the time has zero UTC offset [33].

Table 2-4: General format of weather data.

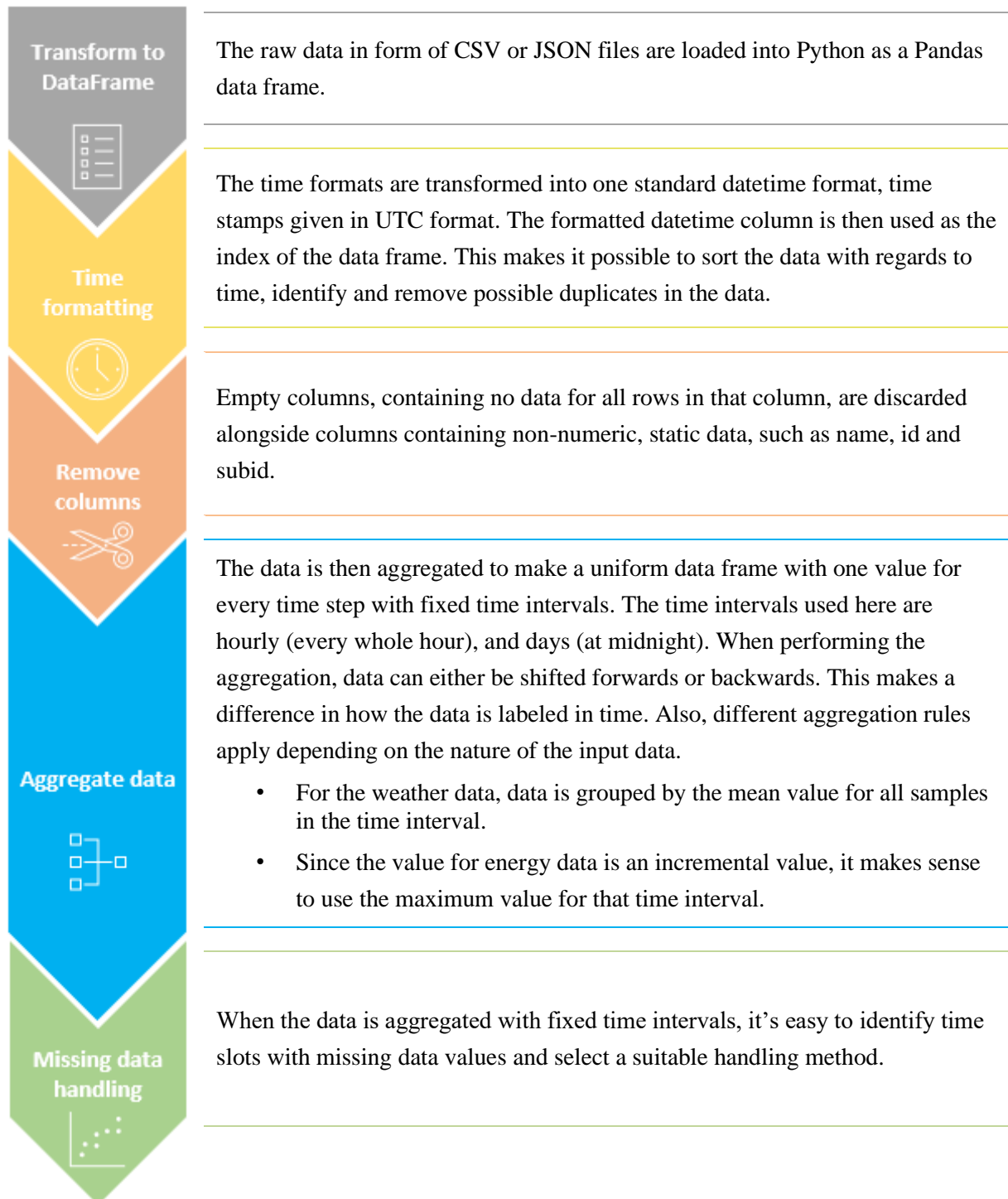
Feature name	Unit	Sample Format	Description
Time (UTC)	Datetime	1577836800000000000	Date timestamp of record in Unix time format ² , nanoseconds
Cloudiness_pct	%	20	Cloudiness in percentage
Humidity	%	59	Relative humidity
Pressure	mbar	1022	Atmospheric pressure
Sunrise_ts	Datetime	1585800496	Datetime for sunrise current day in Unix time format, seconds
Sunset_ts	Datetime	1585846115	Datetime for sunset current day in Unix time format, seconds
Temp_feels_like	°C	5.18	Feels like temperature also known as apparent temperature. The perceived temperature equivalent for humans from combining the temperature, humidity, and wind speed [26].
Temp_min	°C	7.22	Minimum temperature
Temp_max	°C	15.97	Maximum temperature
Temperature	°C	17.81	Air temperature
Wind_deg	deg	38	Wind direction in degrees
Wind_speed	m/s	2.21	Wind speed in wind direction

2.5.2 Pre-processing and aggregation of data.

The raw input data has different time formats, may contain duplicates or have missing records, and the sampling rate differs between the files. So, the raw data needs to be pre-processed and aggregated before it is ready to be further analyzed or used in any ML model.

² Unix (or Unix epoch) time format defines the number of seconds passed since January 1st, 1970, at UTC. A time format commonly used by computer systems [32].

General pre-processing pipeline from raw to transformed data:



As mentioned above in the general pre-processing overview, when performing aggregation data can either be shifted forwards or backwards. This makes a difference in how the data is labeled in time.

- Shifting forwards can be viewed as the value for the previous time slot. Ex. Aggregated data labeled 12:00 will be the values in the time slot from 11:00-11:59. Example shown in Figure 2-21.
- Shifting backwards can be viewed as the value for the coming time slot. Ex. Aggregated data labeled 12:00 will be the time slot from 12:00-12:59. Example shown in Figure 2-22.

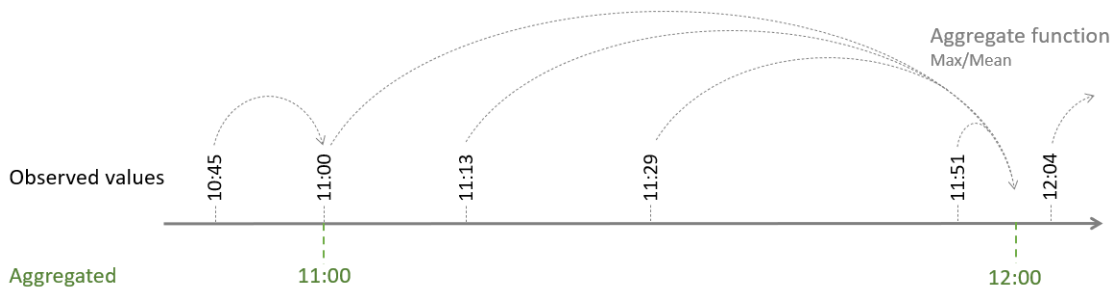


Figure 2-21: Aggregation of data, values shifted forwards.

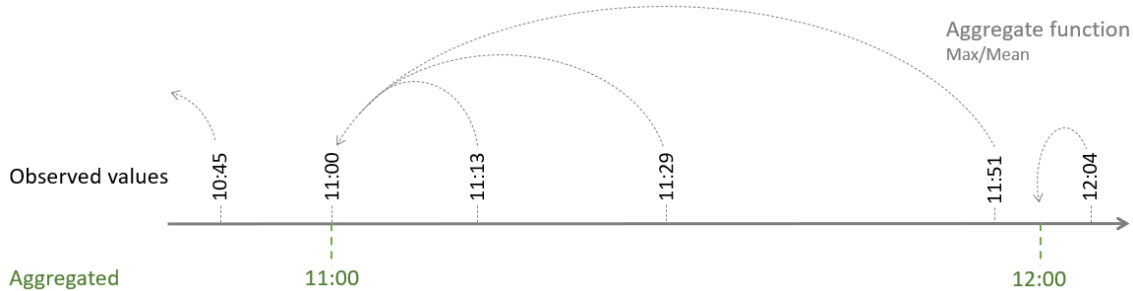


Figure 2-22: Aggregation of data, values shifted backwards.

Data leakage points to the phenomenon where a model is trained on information outside the training data, information that is not known at that point in time. This can make the model perform well on training and validation sets but have reduced prediction accuracy in the real world as these features are not available. Using aggregation with shifting backward rule can be more intuitive to interpret, but it can also make a logical problem when modelling as for example at exactly 12:00 o'clock we cannot know the values that come in the future (12:00-12:59). Aggregation with forward shifting values is therefore used in this project to avoid the possible data leakage.

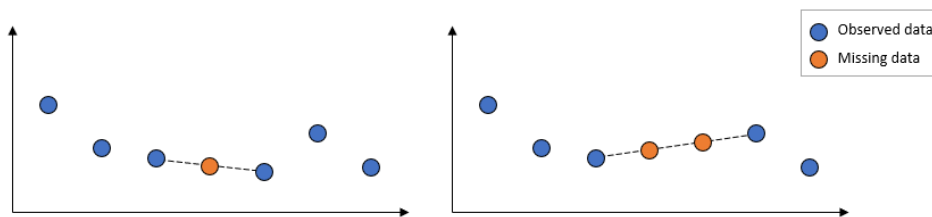


Figure 2-23: Linear interpolation for missing data, for one and several samples missing.

2 Methodology

There are several options when it comes to handling missing values in the aggregated data. The simplest solution would be to do nothing or just drop the rows with missing samples. Another way is to use an interpolation method and substitute the missing values, one of which is linear interpolation illustrated in Figure 2-23. Some other options are:

- Polynomial interpolation.
- Insert mean.
- Insert zero or other fixed value.
- Forward or backward fill, copy previous or next sample value.
- K-nearest neighbor (KNN).

In this project, small gaps of missing data are substituted through linear interpolation. Bigger gaps (many days) of missing data are left unchanged as it is hard to substitute sensible values for these areas.

2.5.3 Time stamps

UTC (Universal Time Coordinated or Coordinated Universal Time) is the world's primary standard for regulating clocks and time. UTC is a measure of mean solar time at longitude 0° with each day being 24 hours and is not adjusted for daylight saving time, 12 o'clock is when the sun is passing the prime meridian. UTC is an efficient replacement for the GTM (Greenwich Mean Time) [27].

For convenient reasons the earth is divided into time zones, where one time zone covers approximately 15 degrees longitude, but time zones tend to follow country borders and not strictly the longitude. All time zones are defined with an UTC offset, usually a whole number of hours, ranging from -12 to +14 hours [28]. For instance, Norway is in time zone UTC+1 and Greece is in time zone UTC+2. Each time zone then uses this UTC time with offset as the local time, so that the sun is at the highest point in this time zone around noon local time.

Many of the countries with higher latitude (the northern countries, but also some of the southern), practice daylight savings time (DST). Daylight savings time shifts the local time (most commonly with one hour) during the warmer months so that darkness comes at a later point in time [29]. For instance, Norway will use UTC+2 instead of UTC+1 and Greece will use UTC+3 instead of UTC+2 during summertime. In the EU, DST begins the last Sunday of March and ends the last Sunday in October at 1:00 UTC time.

All the input data used in this project is marked with a time stamp in UTC+0, this makes it easy to compare data from various sources. One thing to consider is that the local time is changed due to DST, which can also result in a one-hour shift in energy usage pattern during summertime.

2.6 Feature Engineering

In order to help the models perform better, new features can be created from existing data in a process known as *feature engineering*. The core of feature engineering is simply making the data better suited for the task at hand, and consequently adding more value to the data. One example of feature engineering could be the creation of apparent temperature (“felt like temperature”) by combining the values of temperature, humidity, and wind speed [27]. Using

Fourier transformation for picking up frequency components in the data can be another example, which is also a common feature engineering approach wherever this is appropriate [12]. The actual usefulness of a feature will depend on the model, the only useful features are the ones that the model can learn. Sometimes a feature only adds value when in combination with others.

2.6.1 Weather data

As mentioned in the pre-processing and aggregation section (2.5.2), the aggregation rule applied to numerical weather data is the mean value for the aggregation period. But the raw weather dataset also contains datetime values for sunset and sunrise. When aggregated for days, a numerical value for the number of hours with sunlight in that day is calculated based on these two values and added as a new column (*Daylight*). When aggregated for hours these values are discarded.

2.6.2 Energy consumption

The energy data is provided as an energy meter value, and this value does not give us much information on its own as this is only an incremental value for the total energy consumption during the energy meter's lifetime. The value we are interested in is the energy consumption for that specific time period (per hour or day). In order to find the energy consumption for this specific time period, the differential value between two time steps is calculated (as shown in Figure 2-24).

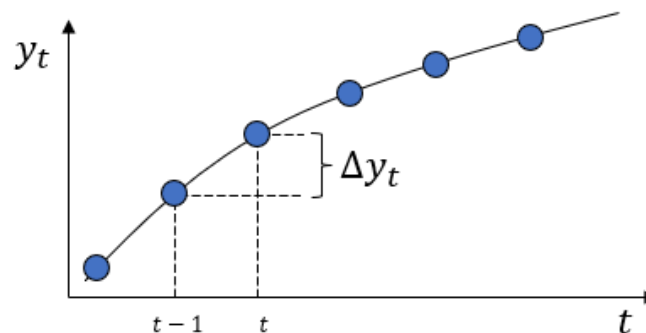


Figure 2-24: Differential value for one time slot

Therefore, a new column representing this differential value is created in all the aggregated energy datasets (one for each energy meter). This differential value is then calculated by taking the current meter value and subtracting the previous. Figure 2-25 shows an example of how this calculation is performed.

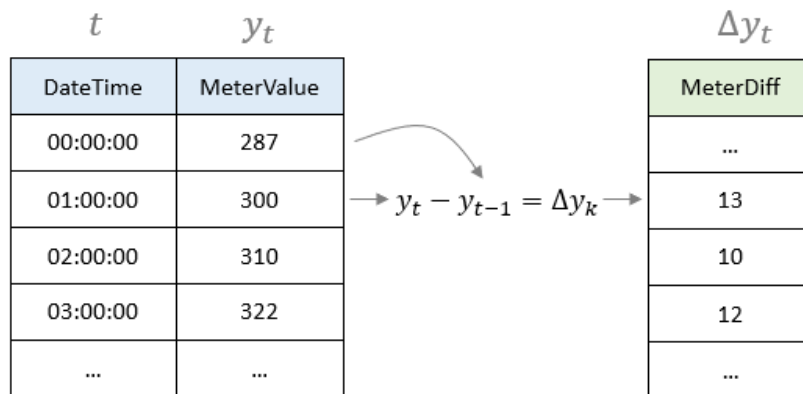


Figure 2-25: Example of how energy consumption is calculated and stored in a new column (*MeterDiff*).

To get the total energy consumption within one building, the calculated energy consumption from all energy meters (*MeterDiff*) in that building is summarized into one parameter named *SumDiff*.

2.6.3 Time Features

For adding timely information, new time related features are created based on the timestamp of each sample. The features created from timestamp are:

- Hour of day (0-23)
- Weekday/day of week (0-6)
- Week number (1-53)
- Month (1-12)
- Day of month (1-31)
- Quarter (1-4)
- Year
- Day of year (1-366)

By doing so, samples can then easily be grouped on various time features during analysis or visualization. The models can also benefit from this information as this can help extract timely information from the data. One thing to consider here is whether these timely features, and particularly hour of day, should be extracted from UTC or local time.

Unlike other numerical values (like temperature) these created timely features are more of a categorical variable, as the magnitude of the value has nothing to do with the variable magnitude. Example: Either it is Monday, or it is not (1 or 0). One option could therefore be to transform these variables with one-hot encoding which usually yields better performance for some model types. The one-hot encoding of time features:

- *IsWeekend* (Mon-Fri = 0 and Sat-Sun = 1)
- *IsWorkHour* (8 <= hour <= 20 then 1, else 0)
- *IsMonday*, *IsTuesday*, *IsWednesday*, *IsThursday*, *IsFriday*, *IsSaturday* and *IsSunday* (true or false, depending on the day of week).

2.6.4 Lag Features

For utilizing the energy data several *lag* features can be created. *Lag* addresses the point that these events are “lagging” behind the current event in time, they happened in the past, as opposed to a lead feature which would be into the future, “leading” in time. How to select what lag features to create will depend on cycles and seasonality in the data. It is reasonable to believe that energy consumption data will be in daily, weekly, and yearly cycles.

Longer lag features created are:

- *Lag1*: 364 days
 - Practically one year back (365 days), but gets the same day of week by using 52 weeks back ($52 \times 7 = 364$)
- *Lag2*: 728 days
- *Lag3*: 1092 days

Shorter lag features are:

- *slag1*: 1 days
- *slag7*: 7 days
- *slag14*: 14 days
- *slag21*: 21 days

These lag features can be used as inputs to help the model by supplying the energy consumption for the same time slot some time step earlier. How far into the past this time step should be will depend on how far the prediction horizon of the model needs to be, as this will limit the maximum prediction horizon. For example if the model’s prediction horizon is seven days, the shortest possible lag factor used must be the same (seven days back). Using *slag1* in this case would create a violation on prediction for the second day, as this information is not known. It is therefore important when using the lag features as model inputs to take particular care to avoid such information violations.

2.7 Data Analysis

For several reasons (like reduced training time and risk for overfitting) it is desirable to keep the models as simple as possible. It is therefore desirable to identify correlation between variables, identify if they contain duplicate information, and in turn uncover the variables that are influencing the model’s target value the most. As a result, the model can use only the inputs adding substantial value and the others providing mostly noise be discarded.

The aggregated data is therefore inspected and analyzed to get a basic understanding of the data content and system dynamics. Several plots are created for visual inspection and simple calculations like correlation and autocorrelation are used as tools for identifying the important variables.

2.7.1 Trend

The underlying trend for the data is calculated as the simple 365-days moving average (SMA). For the first year (day 1-365) the trend is set equal to this period’s mean value as this year must

be used to initialize the SMA (horizontal trend first year). The first year then sets a baseline value and a trend factor can be calculated by dividing the SMA output on the baseline value.

2.8 Modelling

The objective for all models in this thesis is to forecast energy consumption in a building some time step into the future. Since the target of the models is a numerical value, this makes it a regression problem.

There are several methods for setting up the model structure, this thesis will focus on two different approaches, with and without the use of sequences of input data. Figure 2-26 illustrates the difference between the two approaches. Where the one to the left uses inputs in the form of a sequence to forecast the energy consumption some amount of time into the future. The length of the sequence (number of time steps back) and the number of variables used can vary, in addition to the prediction horizon (time steps k into future). With this approach it is typical that the target variable (output) is also used as an input variable as the model tries to forecast the next value in this sequence.

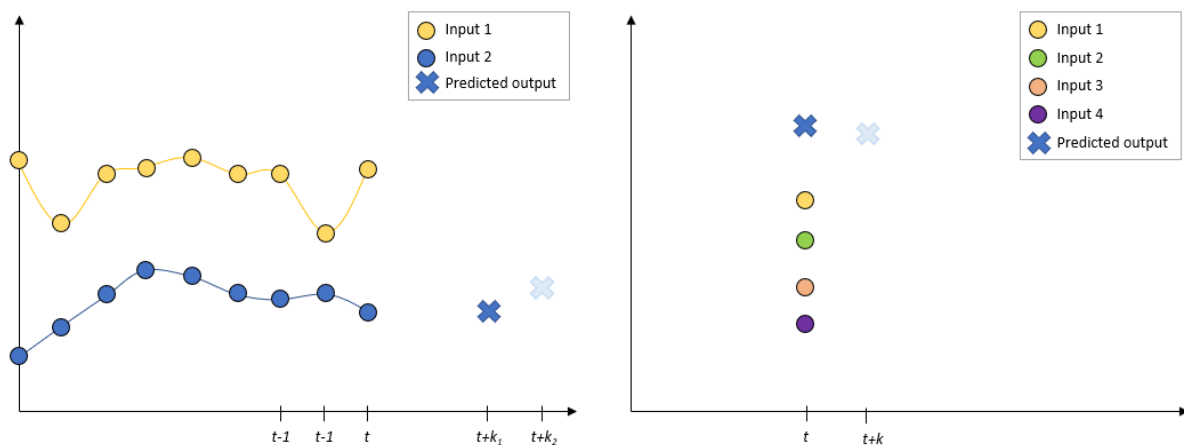


Figure 2-26: Different model prediction approaches.

The other approach (to the right in figure) does not use an input sequence, but rather simple, static inputs. As opposed to the previous approach, it is not common to also use the target variable as an input in this approach, as this model tries to predict this missing value. The example in the figure predicts the target value at the same time slot as the input variables, but it could also have been some other time slot into the future. Based on the inputs the model will try to predict the corresponding output, time is not important in this approach.

It can be reasonable to believe that a sequence of input data is more important if the prediction horizon is short, as the sequence can then carry relevant information from the near past (example: for predicting the energy consumption for the next day, the pattern from the previous seven days can be important). But as the prediction horizon gets larger (week or even month), the gap between the data available and the target value becomes larger and thus increasingly uncorrelated, and a model with static inputs can be just as good as using an input sequence.

In this project two types of ML models are considered, extreme gradient boosting and neural networks (LSTM), as these are lately one of the most used algorithms for a wide selection of

problems. Both methods are described in more detail in the following section. The variables used as inputs and the general structure of the models will depend on the findings from the data analysis. In any case, it is normal to start with a simple model structure and only add complexity if it improves the predictions in order to avoid unnecessary complex models.

2.8.1 Extreme gradient boosting

Since decision trees or gradient boosting machines are not capable of handling sequences of inputs, the second approach with simple, static inputs is used for this model type. The library XGBoost is used for implementing these models in Python.

The model's maximum prediction horizon will depend on the inputs used to the model. For instance, if only timely features are used as an input (quarter, month, day of week...etc.), the prediction horizon is infinite as we can produce these inputs for the infinite future. But if we also include features such as lag or weather features into the model input, the prediction horizon will be limited by the extent of how long into the future these variables are known.

To prevent overfitting and keep the model as simple as possible, it is best to start with a simple model (few, shallow trees with few inputs) and keep adding complexity until the validation set stops improving. The parameter for early stopping is also used during all model training as a tool to help against overfitting, so that the training stops when the validation set stops improving for the given number of steps.

2.8.2 LSTM

LSTM is the neural network type selected in this project because of the ability to handle sequences of timeseries as input data. LSTM networks are implemented in Python by using the TensorFlow and Keras libraries. When working with LSTM network and timeseries data, the data first needs to be in a format the neural network can understand. So, the timeseries data is divided into small input sequences and the corresponding target sequence in a sliding window style. Figure 2-27 shows how one such input and corresponding labeled target sequence is organized.

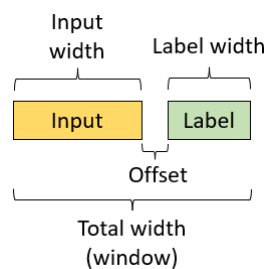


Figure 2-27: Input and labeled target sequence to LSTM.

These small sequences of input and target values are then stacked into a *batch*. Figure 2-28 illustrates one example of this batching principle for one timeseries, where a 3-sample input sequence and the corresponding target with two samples ahead (offset one) is displayed.

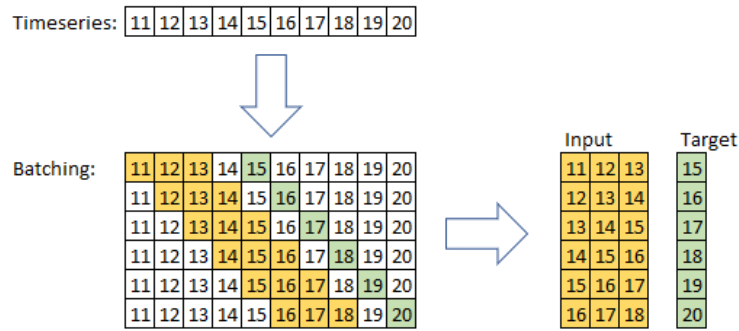


Figure 2-28: Example of batching timeseries data for training the LSTM network.

It is not unusual to have more than one input to the LSTM network, Figure 2-29 shows how this same batching principle will work for multiple inputs and one output. These kinds of data structures are known as *tensors* in TensorFlow and Keras.

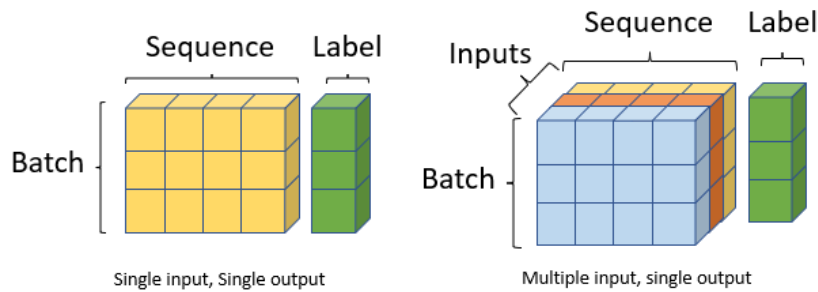


Figure 2-29: Concept of batching timeseries data (rank-2 and rank-3 tensors).

When dividing the data into training, validation and test set the focus in this case will be on the target values. Example: the corresponding input sequence to the first target value in the test set will be indexed in the validation set. Figure 2-30 shows an example of how this will affect the indexing of input and outputs for the training, validation, and test set where an input sequence width of seven, the offset is zero and target width is one.

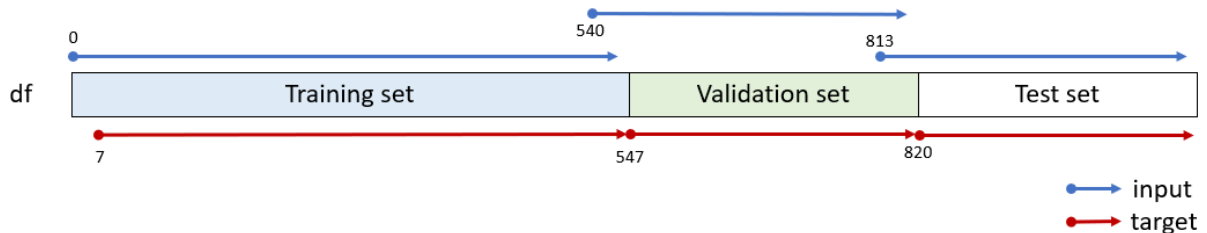


Figure 2-30: Example of indexing of training, validation, and test set. Starting index for input and output matrices.

Since the LSTM models are initialized with random weights, the model training result can be different each time. Accordingly, it can be beneficial to use several training rounds per hyperparameter setting in order to test them appropriately. Since it can be hard to exactly replicate a model's training outcome, it can be convenient to save good performing models for later import and reuse.

2.8.3 Validation of models

MAE (Mean Absolute Error) and RMSE (Root Mean Squared Error) are two metric measures used for validating the model performance. Here, the MAE value is used to determine the best performing models. Cross validation could be used, crucial when there is little data. Cross validation is not as important when having a sufficient amount of data, as in this project.

2.8.4 Data standardization

Some model types, like neural networks, are particularly sensitive to the different scales and variances that naturally come with different variables and units. In order for every variable to have the same potential impact effect on the model, it is normal to standardize variables to mean and variance. One common way is to center to mean and scale to standard deviations from mean, also known as z-score standardization. Equation (2.3) shows the formula for z-score standardization.

$$z_{scaled} = \frac{x - \bar{x}}{\sigma_x} \quad (2.3)$$

Another way can be to scale to feature range, then features are scaled to lie between a minimum and maximum value (ex: 0 and 1). This can be achieved by max-min scaling.

2.8.5 One-hot encoding

Neural networks can be more sensitive to categorical inputs compared to decision trees. Then instead of using the variable *DayOfWeek* as input, it can then make more sense to use some of the variables *IsMonday*, *IsTuesday* and so on. Principle of one-hot encoding is described in chapter for time feature engineering (chap. 2.6.3).

2.8.6 Outlier handling

An outlier is a sample that deviates significantly from other observations in the same dataset. Outliers are introduced from several causes, in this particular dataset there could for instance be errors with a sensor, equipment failure, grid power fails, data processing error and so on. But there could also be abnormalities in the building, something that went wrong, someone opened a window and so on. Before the model training can begin, it is important to handle outliers in the training data so that the models do not try to also learn these outliers. How to correctly handle the outliers can depend on the reasons behind the outlier, but this is normally unknown. The simplest way to handle an outlier can be to just remove the values including outliers, other times this may not be appropriate, then substituting the outlier value for another value can be a choice (same principle as for substituting other missing data). There are several known methods for identifying outliers or abnormalities. Four of which are:

- Statistical methods like Interquartile Range (IQR) and standard deviation.
- Hampel filter [30].
- STL (Seasonal-Trend decomposition using Loess)
- Automatic outlier detection methods: isolation forest, one class SVM, elliptic envelope and local outlier factors.

3 Data analysis results

This chapter presents the result of data pre-processing and analysis of the data performed in Python. The result of the pre-processing data pipeline is several data frames:

- Weather data
 - One aggregated for hours and one for days.
- Energy data
 - Containing data for each individual energy meter and total sum.
 - One aggregated for hours and one for days - two for each building.
- Combination data
 - Combination of weather and energy data.
 - All weather features, combined with total energy consumption for each building.
 - One aggregated for hours and one for days - two for each building.

3.1 Weather data analysis

This subchapter will focus on the weather data analysis, starting with a basic inspection through line plots. A comment on the correlation for weather features is found as a part of the data analysis of combined data (chap. 3.3.2).

Figure 3-1 shows a line plot of all the variables in the weather dataset, aggregated for days. Here we can see that most of the weather features carry a seasonal tendency.

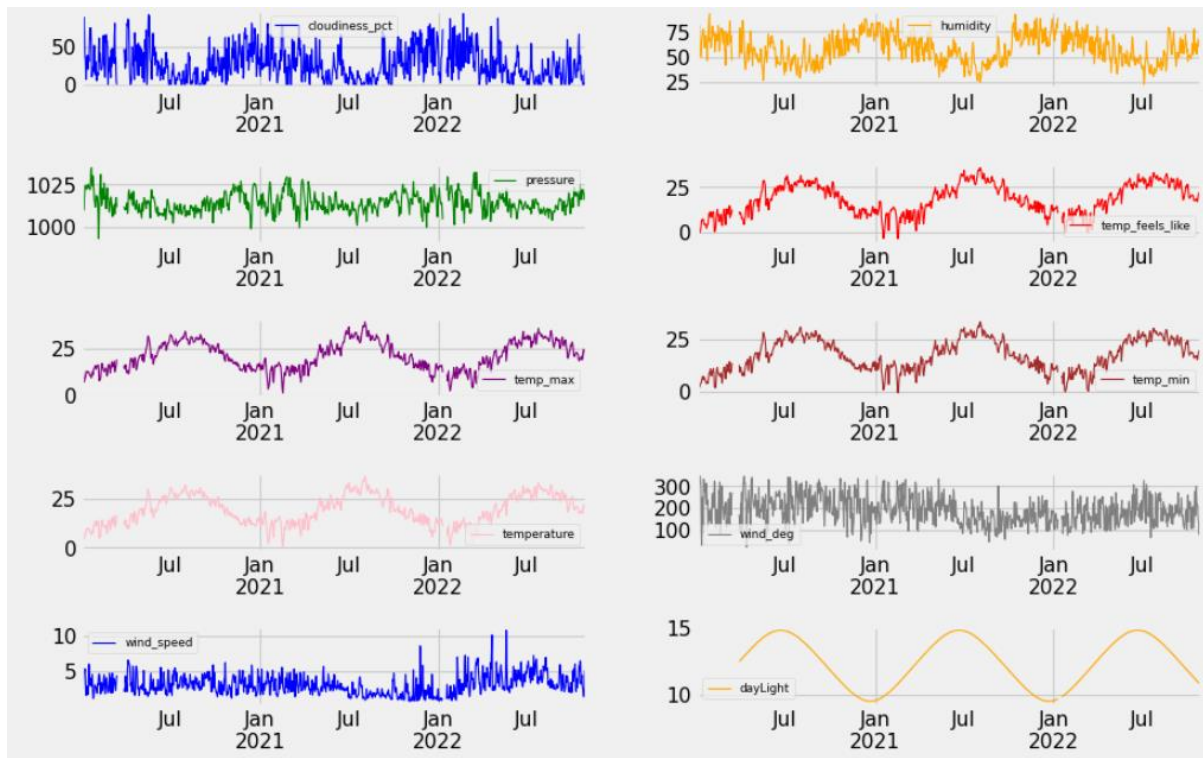


Figure 3-1: All weather data, aggregated for days.

3 Data analysis results

Figure 3-2 shows a line plot of temperature, “feels-like-temperature” and 30 days simple moving average of temperature (SMA30). The figure also indicates the time periods with missing temperature data with a thick red line (at $y=0$). Here, the missing data is not substituted as the gaps with missing data are large (several days).

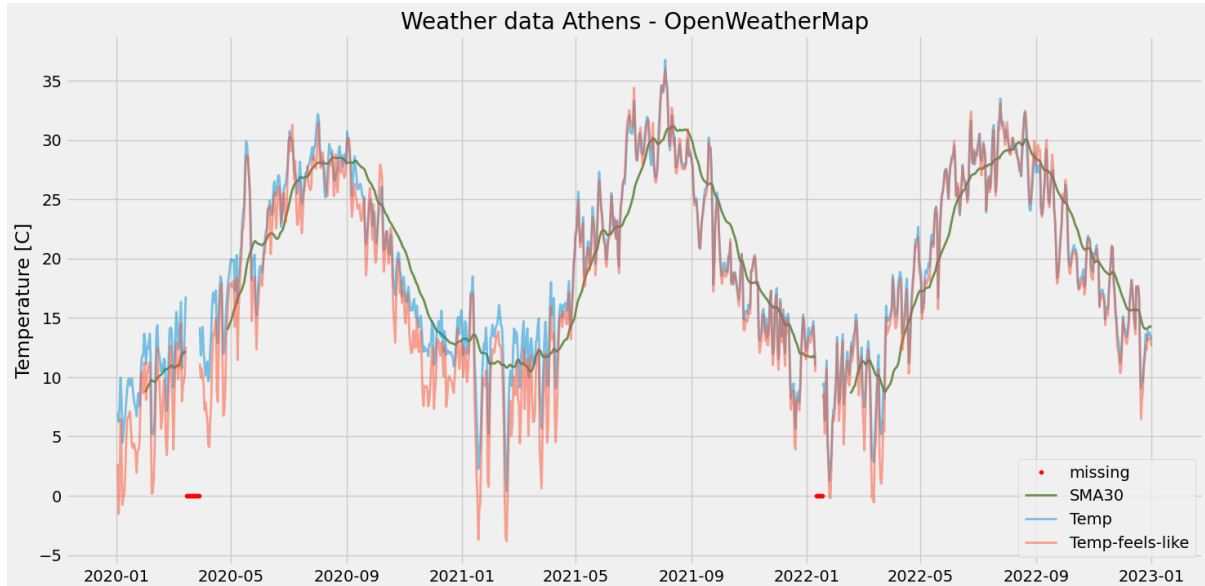


Figure 3-2: Plot of weather data: temperature, felt-like-temp, SMA30. Data aggregated for days.

In this plot it seems that the feels-like-temperature deviates more from the temperature feature when the temperature is in the lower parts of the chart compared to middle and top.

3.2 Energy data analysis

This subchapter will focus on examining the energy data for both buildings. As described in the methodology chapter, the calculated energy consumption variable will be used in the energy data analysis.

3.2.1 Building #1

Figure 3-3 shows a stack plot of all the individual energy meters in building #1, and Figure 3-4 a boxplot of the same energy meters. Here it can be seen that some energy meters contribute far more to the total energy consumption than others. And also that the consumption varies during the seasons.

3 Data analysis results

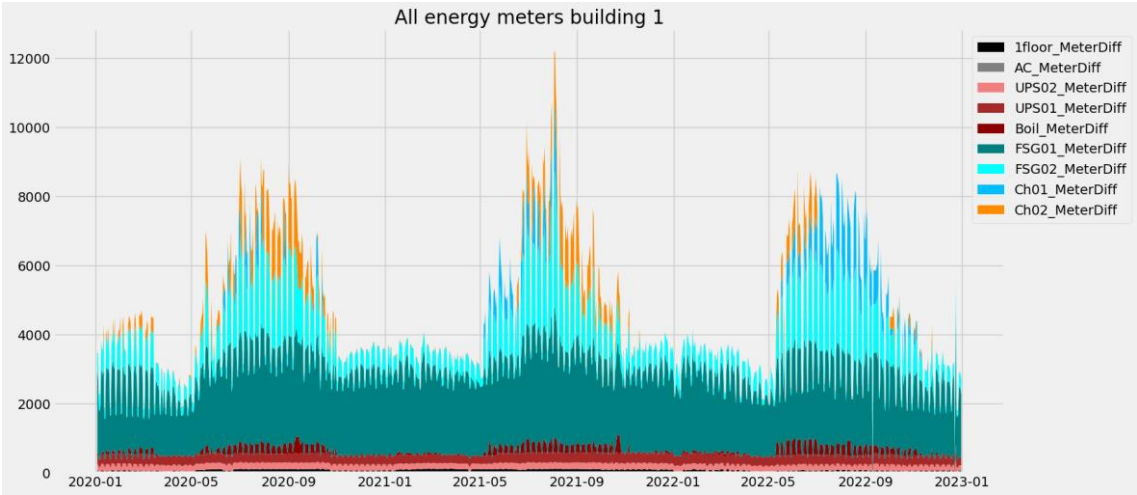


Figure 3-3: Stack plot of energy consumption (aggregated for day) for all energy meters in building #1

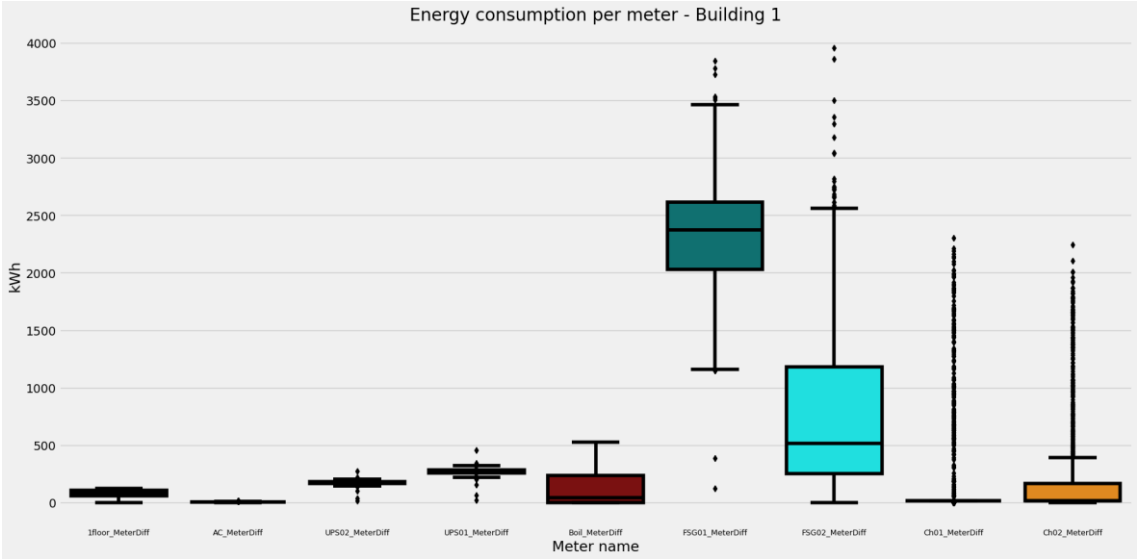


Figure 3-4: Boxplot of energy consumption per day for all energy meters in building #1

3.2.2 Building #2

Figure 3-5 shows a stack plot of all the individual energy meters in building #2, and Figure 3-6 shows a boxplot of the same energy meters. As for building #1, also here some energy meters contribute far more than others. But it doesn't seem to be quite the same percent wise magnitude on the seasonal variations like in building #1. There are however some fluctuations in the data, especially around September 2022, that can indicate some errors in the measurement or data pre-processing.

3 Data analysis results

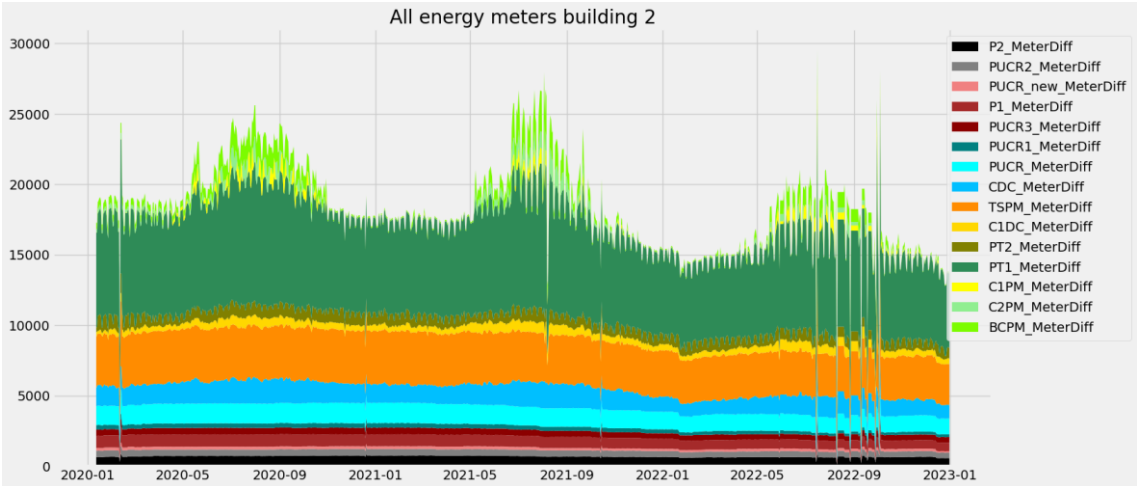


Figure 3-5: Stack plot of energy consumption per day for all energy meters in Building #2.

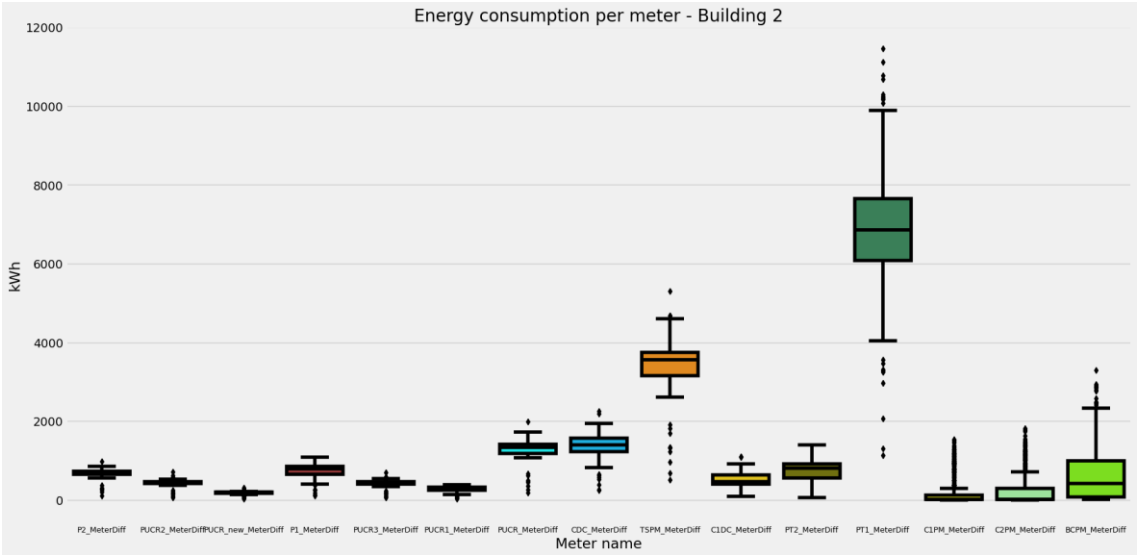


Figure 3-6: Boxplot of energy consumption per day for all energy meters in Building #2

3.2.3 Both buildings

Figure 3-7 shows a line plot of total energy consumption for both buildings. From the mean lines added for each building in the plot, it can be seen that building #2 has a reduction in energy consumption and especially the third year as this period is far below the mean line. It also looks like the energy usage pattern for building #2 has changed during the time period, as there are more alterations during summer months in the last half (start around summer 2021). Even though building #1 has a much larger seasonal increase compared to building #2 (percent wise), it can be seen that both buildings have the same seasonal pattern, with an increase during summer months. This is simpler shown in Figure 3-8 where data is z-score standardized.

3 Data analysis results

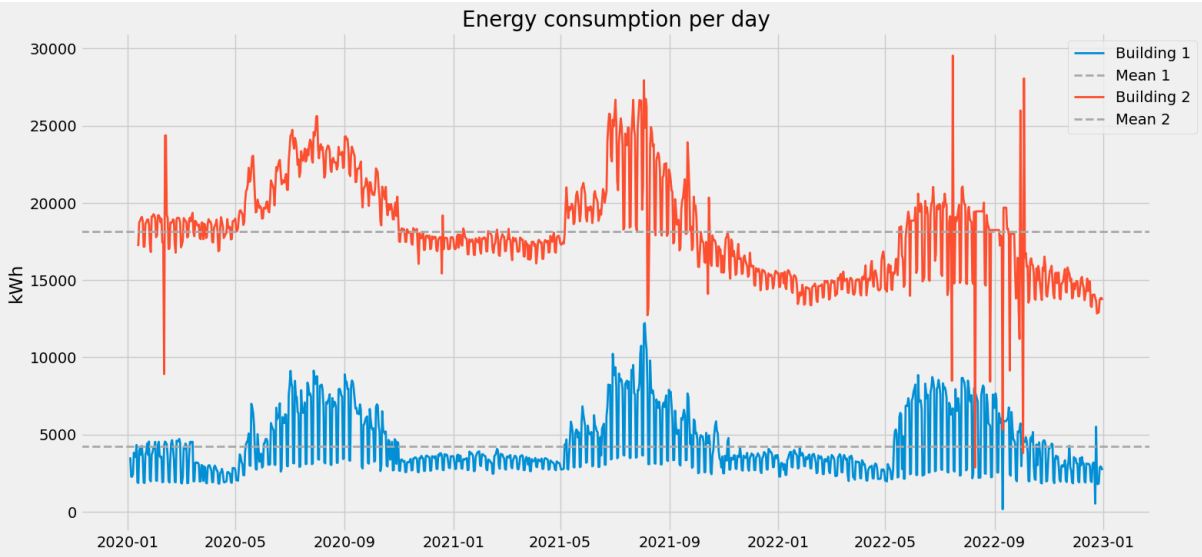


Figure 3-7: Line plot of total energy consumption per building, aggregated for days.

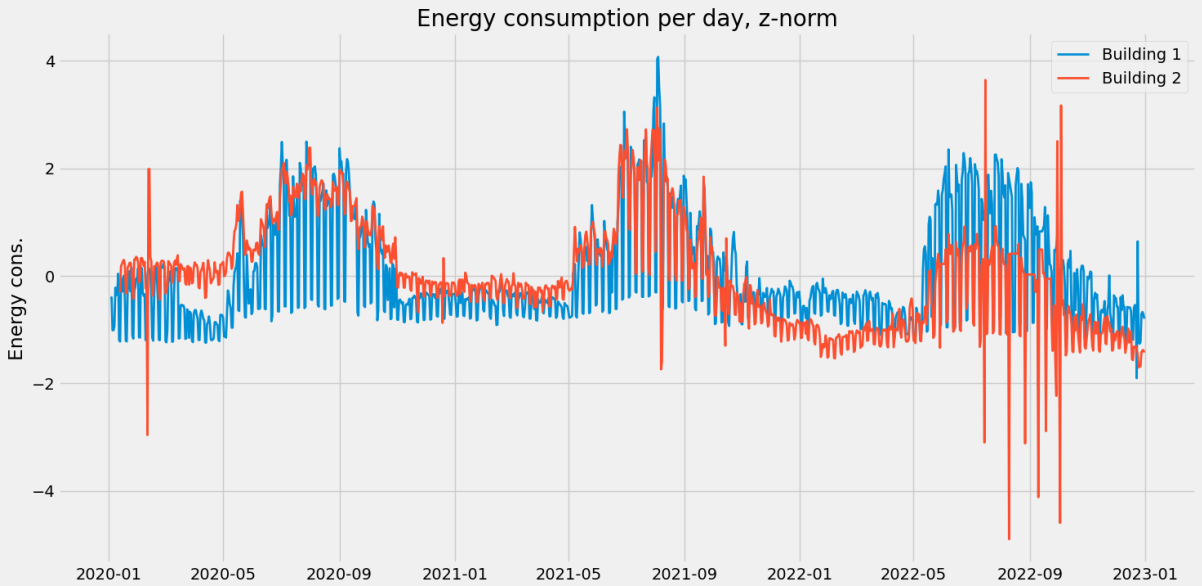


Figure 3-8: Line plot of total energy consumption per building, aggregated for days, z-score standardized data.

3.2.4 Trend

The energy data is divided into separate years for a simpler yearly comparison of the seasonal variations and cycles in the data. Also, the underlying trend in the data is calculated as the 365-day SMA value (as described in chapter 2.7.1). The yearly data for second and third year can then be adjusted for the underlying trend (divide by trend factor) for easier comparison of cyclical behavior in all years.

From the plots in Figure 3-9, we can see that building #1 has no considerable change in the underlying trend and adjusting for trend gives no noticeable change. However, from the plot in Figure 3-10, building #2 has a strong decline in the underlying trend, going from a mean value of 19 970 kWh/day the first year and ending at 15 931 kWh/day the third year, a reduced yearly

3 Data analysis results

mean energy consumption of about 20%. There are of course climate condition factors that could affect the underlying trend, as not every year has the same weather, but such a big decrease would indicate some behavior changes in the energy usage pattern for building #2. Each building has a separate energy sensitivity to climate conditions, meaning the actual percent of energy dependent on climate is different. So, if the changes were purely due to different yearly climate conditions, it would be reasonable to expect a comparable trend in building #1 as both buildings experience more or less the same weather.

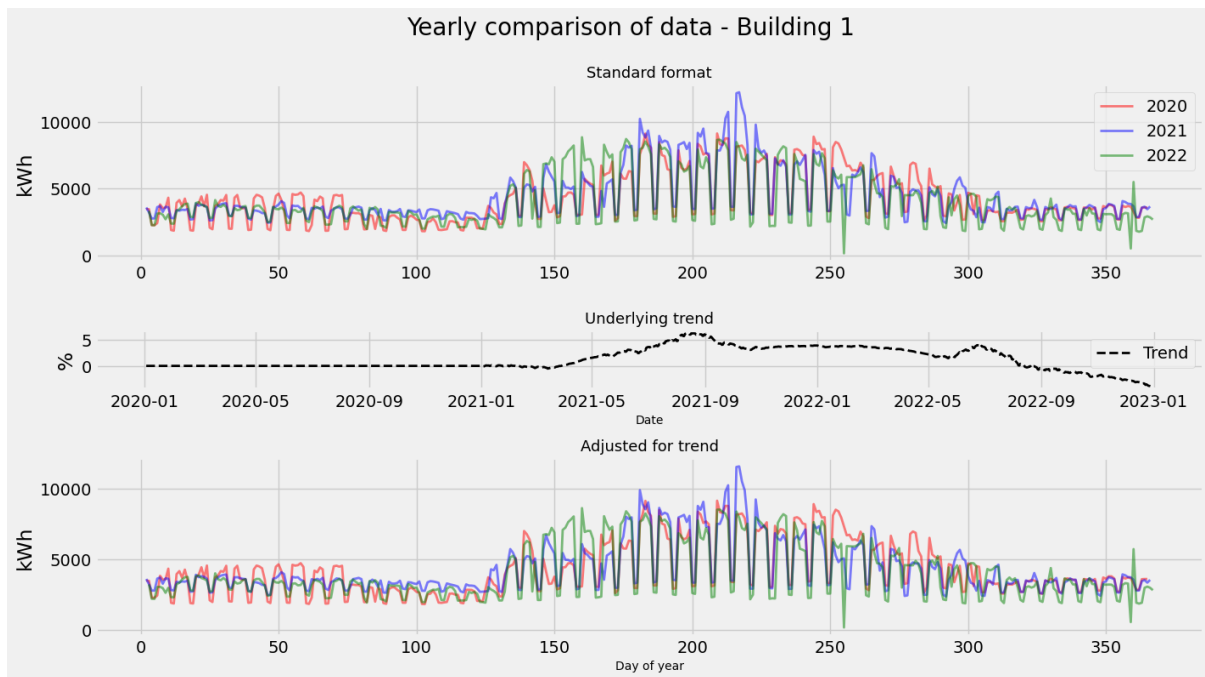


Figure 3-9: Yearly comparison of data, adjusted for leap year and day of week lineup, building #1.

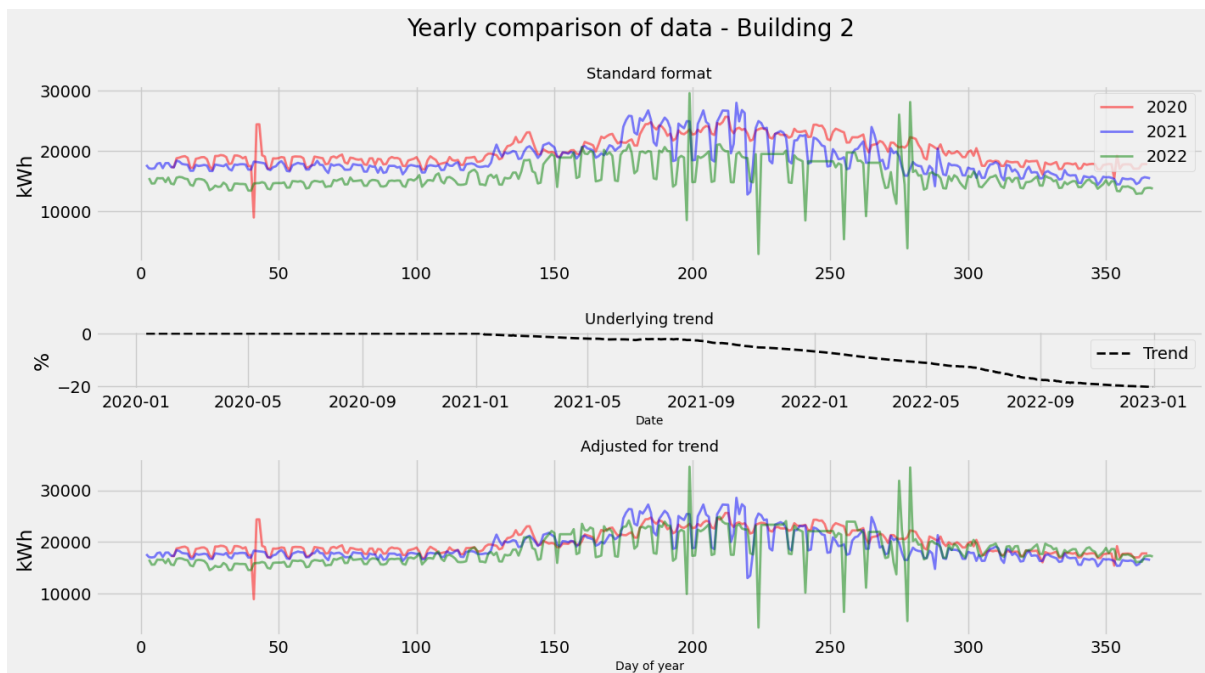


Figure 3-10: Yearly comparison of data, adjusted for leap year and day of week lineup, building #2.

3 Data analysis results

When energy consumption in building #2 is adjusted for the underlying trend (last chart in figure), meaning that the value is divided by the trend factor, the yearly trends lineup fairly well showing a comparable seasonal usage pattern.

3.3 Combined data analysis

It is intuitive that energy consumption in a building is correlated with the outdoor temperature, so for initial examination of this relationship a line plot combining energy consumption with temperature is created (see Figure 3-11). From the figure, it can be seen that the energy consumption for both buildings is highest during the summertime which points to energy consumption used for cooling purposes. Note that the figure contains two y-axes.

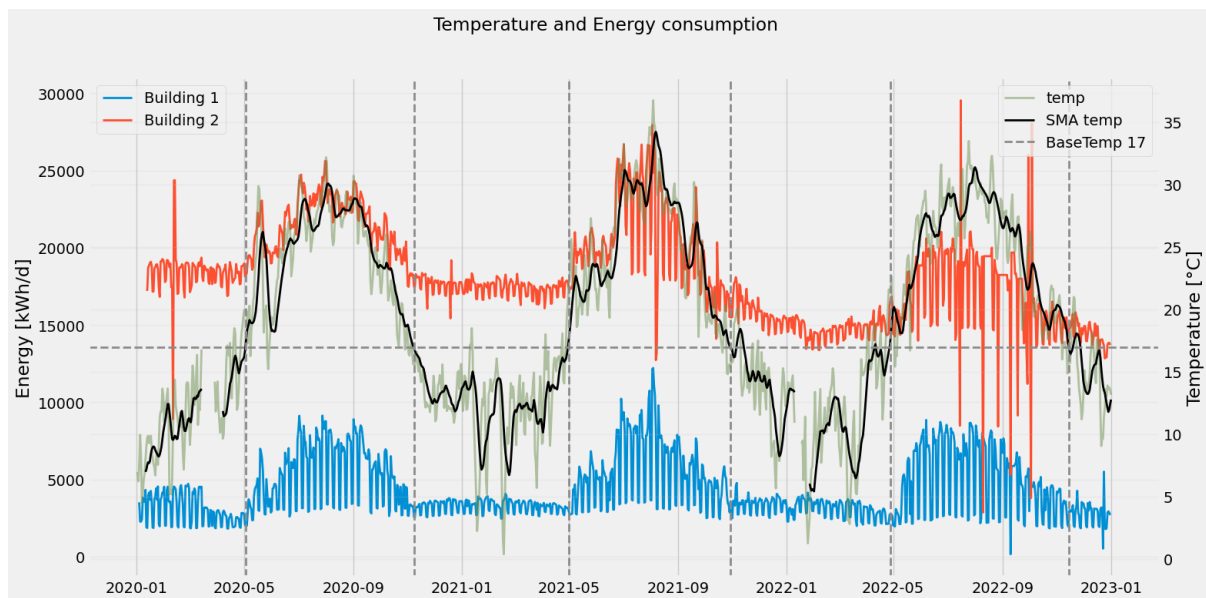


Figure 3-11: Combination plot of energy consumption (left y-axis) and outdoor temperature (right y-axis).

According to Enova (governmental energy advisor organization in Norway), the need for additional heating in buildings normally ends when the outdoor temperature exceeds about 17 degrees Celsius, therefore they name this value *base temperature* [31]. If temperatures below the base temperature require additional heating, it would make sense that values above would call for additional cooling. To investigate this hypothesis, some assisting lines are added in the figure. One horizontal line for the base temperature of 17 degrees, and several vertical lines for the dates when outdoor temperature crosses this base temperature line. Here, a 10-day SMA (simple moving average) value for temperature is used to get a single crossing point of the base temperature. From the plot it seems that these vertical lines correspond well to the points where energy consumption increases and returns down. This indicates that the hypothesis of using 17 as the base temperature could also be applicable for cooling.

3.3.1 Timely energy features

Figure 3-12 and Figure 3-13 shows boxplots of the hourly aggregated data where data is grouped by timely features as hour of day, day of week, month, and year, for both building #1 and #2. For both buildings there seem to be several cyclical behaviors as consumption is higher

3 Data analysis results

during daytime compared to nighttime, higher during weekdays compared to weekdays, and highest in the summertime. Building #2 does also have a descending yearly trend in the data, which is not seen in building #1. Which corresponds to the previous observations and trend calculations.

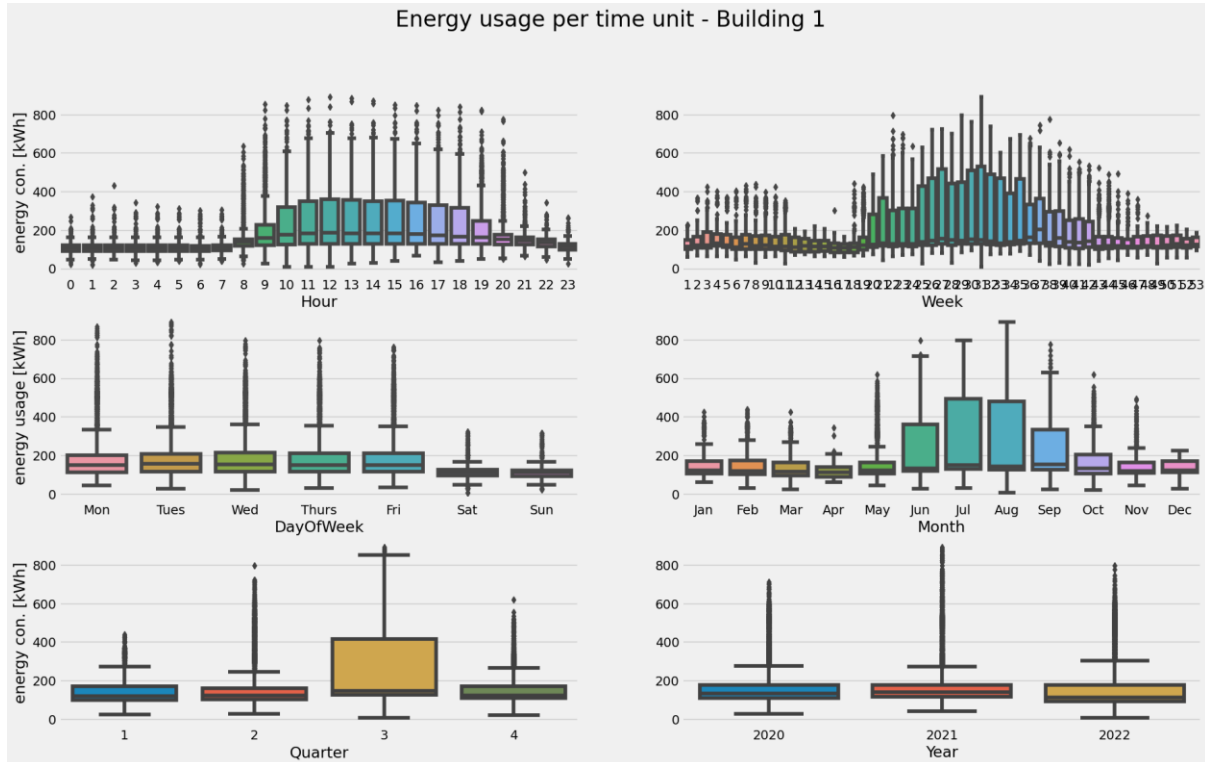


Figure 3-12: Boxplot building #1, hourly aggregated data grouped by time features. Adjusted to local time with DLS.

3 Data analysis results

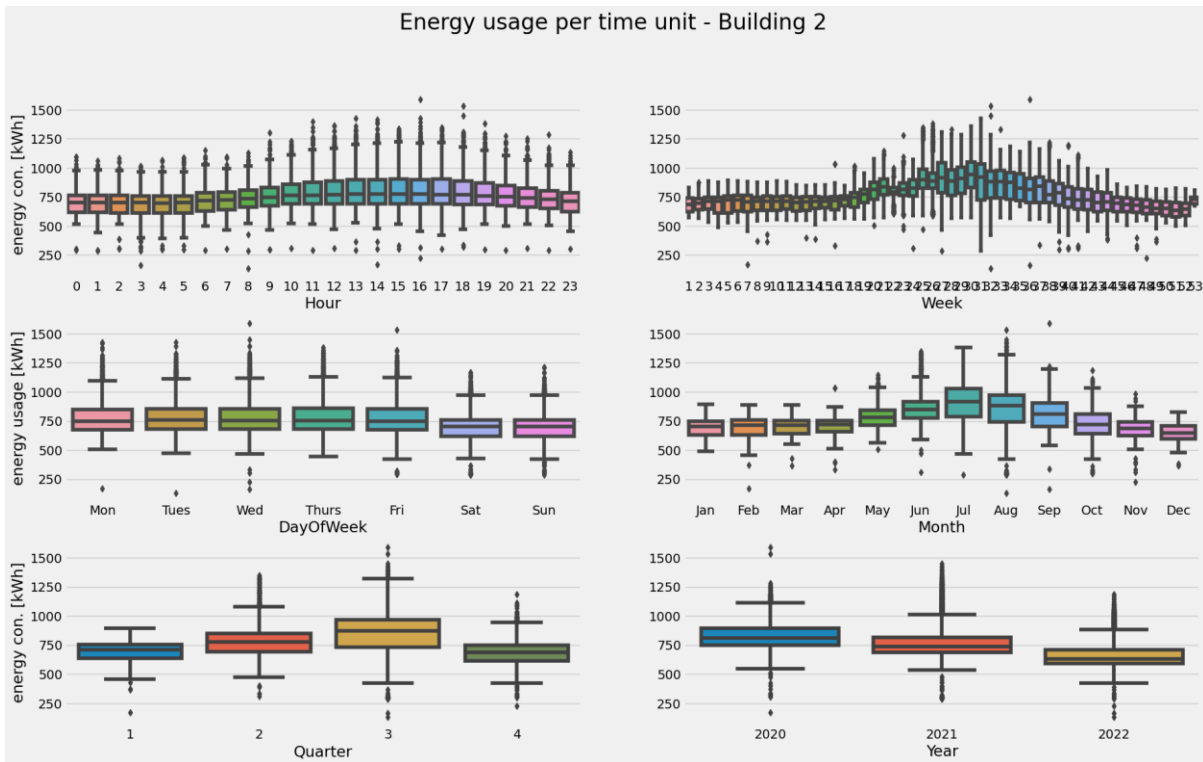


Figure 3-13: Boxplot building #2, hourly aggregated data grouped by time features. Adjusted to local time with DLS.

In the previous boxplots, time features are created from the local time (adjusted for daylight savings time). To illustrate the effect of DLS in the usage pattern, Figure 3-14 shows a comparison where data is divided by summer and winter time (DLS), where one side is adjusted for DLS and the other is not. Here one can see that there is a one-hour shift in the pattern if data is not adjusted for DLS, energy consumption rises one hour earlier when not adjusted during summertime (plot in down left corner).

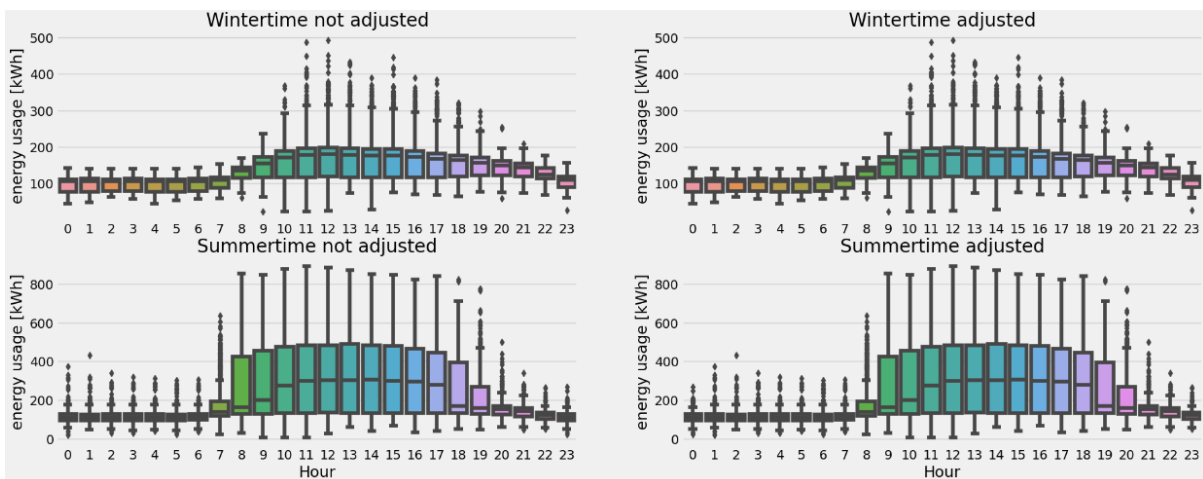


Figure 3-14: Comparison of summer and wintertime, with and without adjusting for DLS. Data for building #1.

3.3.2 Correlation

This subchapter will describe the analysis of variables and the correlation between them. Figure 3-15 and Figure 3-16 shows the correlation of every weather feature combined with energy consumption in building #1 and #2 in triangular heatmaps. From these figures we can see that the variable *SumDiff* mostly correlates with *temperature*, *humidity*, *daylight*, and *cloudiness*, and with the strongest positive correlation between *SumDiff* and *temperature*.

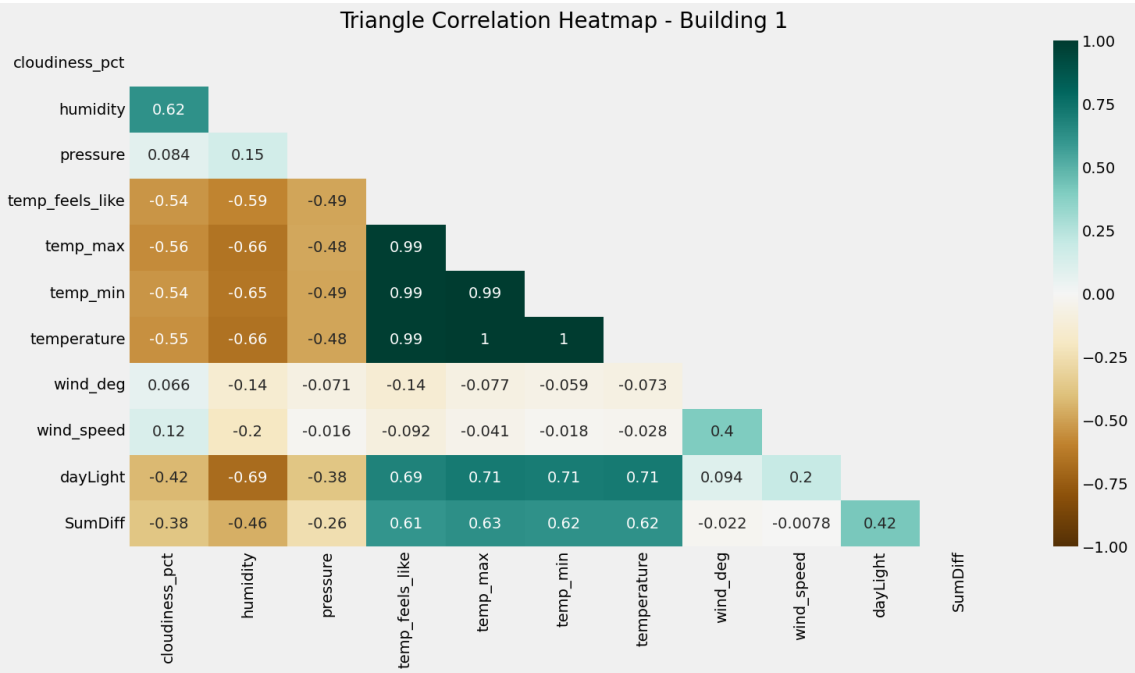


Figure 3-15: Heatmap weather data and *SumDiff* building #1. Aggregated for days.

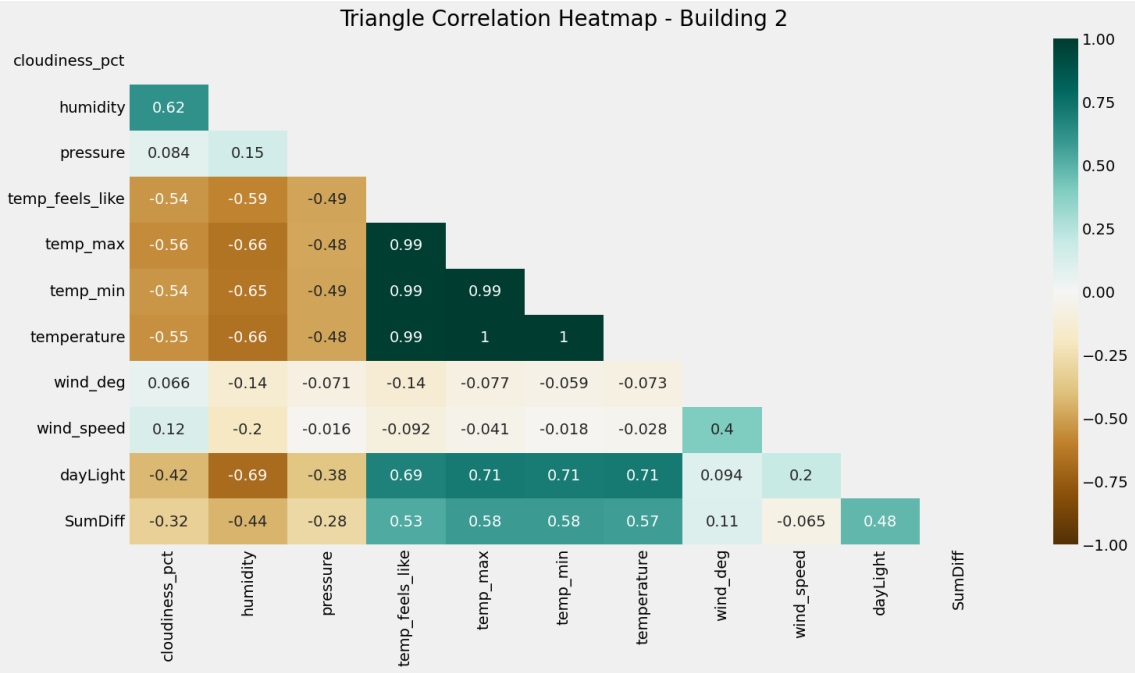


Figure 3-16: Heatmap weather data and *SumDiff* building #2. Aggregated for days.

3 Data analysis results

The most important results from the heatmaps and correlation are summarized in Table 3-1.

Table 3-1: Comments to important feature correlation

Feature	Corr. with <i>SumDiff</i> (Building #1 / Building #2)	Comment
<i>Temperature</i>	0,62 / 0,57	We can see that all temperature variables (<i>Temp_feels_like</i> , <i>Temp_max</i> , <i>Temp_min</i> , and <i>Temperature</i>), has a correlation of approximately 1 meaning these carries mostly the same information. And it would therefore make sense to only use one of them further.
<i>Humidity</i>	-0.46 / -0.44	The relative humidity has an inverse correlation with <i>temperature</i> (-0.66) and <i>Daylight</i> (-0.69). Indicating that the climate is relative more humid with low temperatures and shorter days - wintertime.
<i>Daylight</i>	0.42 / 0.48	Daylight is strongly correlated with temperature (0.71), indicating the carry much of the same information. As the days gets longer, the temperature gets higher.
<i>Cloudiness</i>	-0.38 / -0.32	Negative correlation: when less clouds (open sky) – higher energy consumption. Cloudiness correlates to humidity and temperature, indicating that there are more clouds when RH is high, and temperatures are low.

Based on the previously shown heatmaps and correlation analysis, it is found that *temperature*, *humidity*, *daylight*, and *cloudiness* are the features with greatest correlation with energy consumption (*SumDiff*). Therefore, scatter plots of these variables (x-axis) together with energy consumption (y-axis) are shown in Figure 3-17 and Figure 3-18. From the previous boxplots, where energy is grouped on timely energy features, we know that the energy consumption is dependent on day of week, so the samples are grouped, and color-coded by day of week.

3 Data analysis results

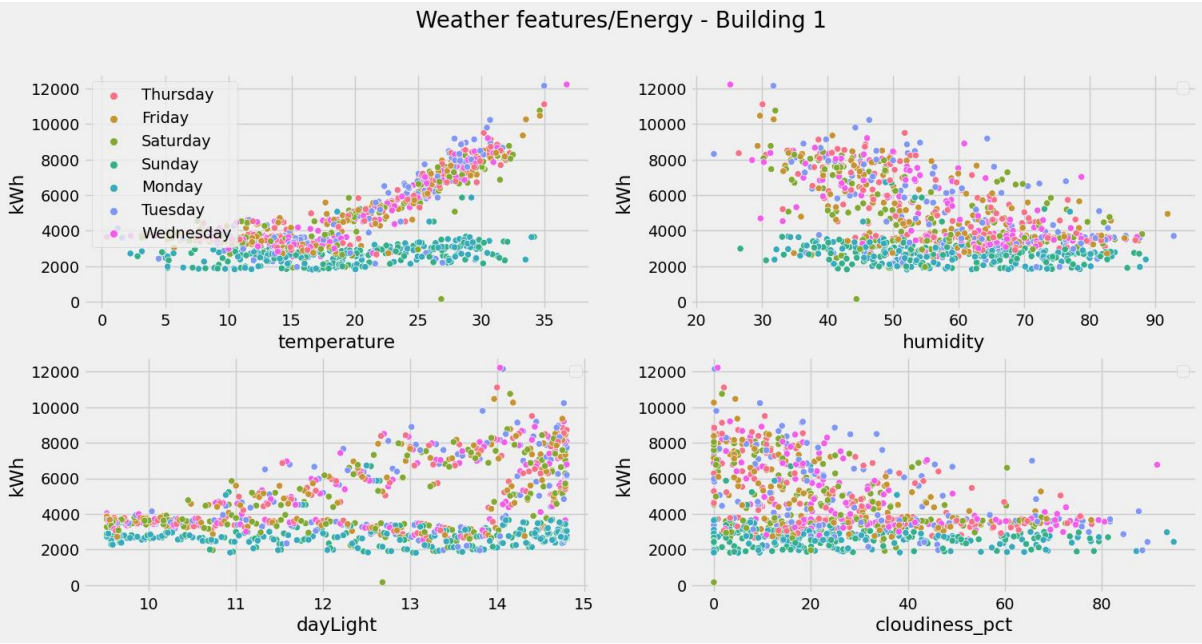


Figure 3-17: Scatter plot of most important weather feature for building #1, aggregated for days.

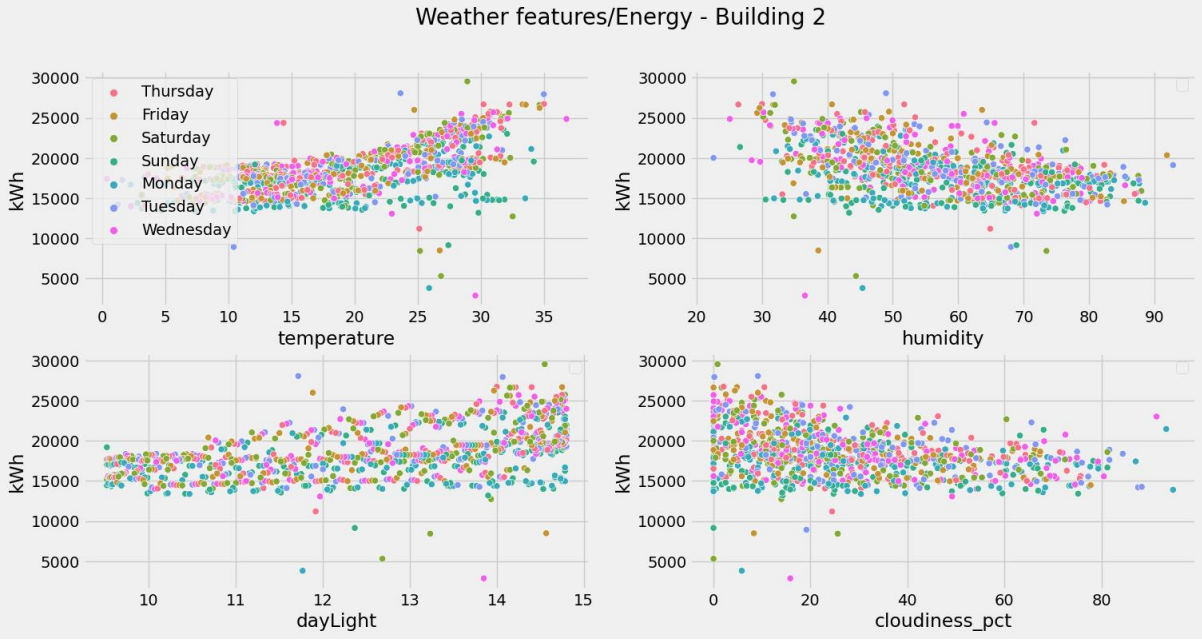


Figure 3-18: Scatter plot of most important weather features for building #2, aggregated for days.

3.3.3 Autocorrelation

Since the lag features are created from the variable *SumDiff*, only with a time difference, it is possible to check the *SumDiff* variable's autocorrelation. The autocorrelation between lag features and *SumDiff* can be seen in Figure 3-19. It shows that building #1 has the strongest autocorrelation to 7 days back of 0.86 (*slag7*) and 52 weeks ago of 0.84 (*lag1*).

3 Data analysis results

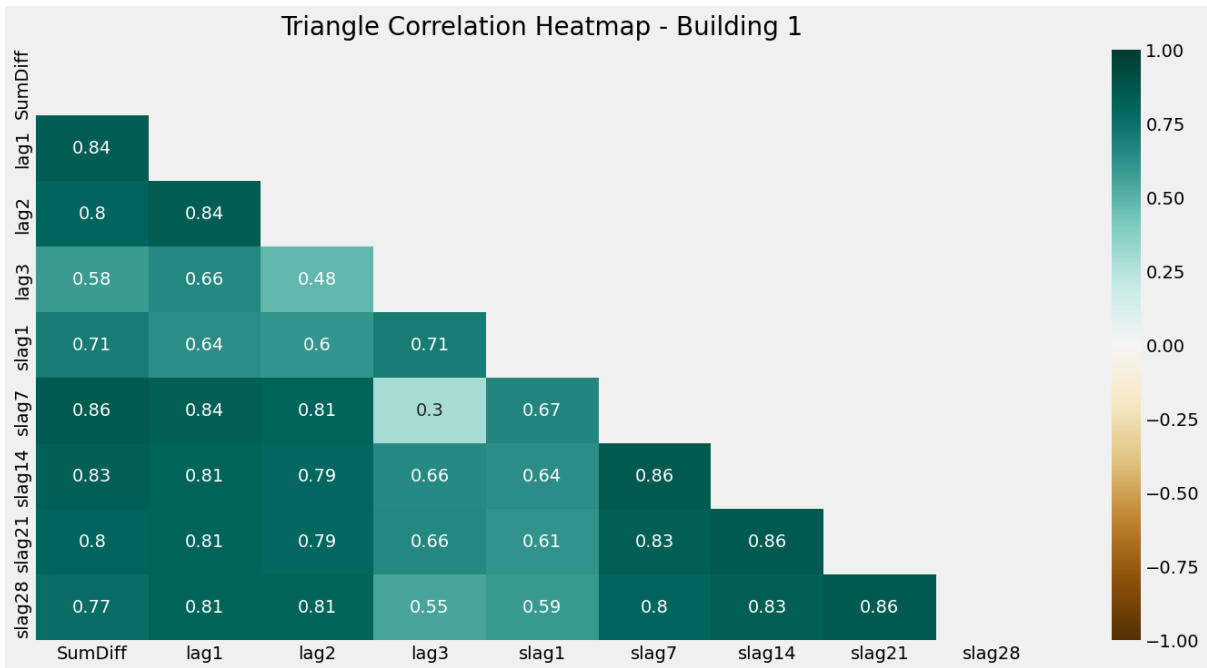


Figure 3-19: Correlation of lag features, building #1.

Figure 3-20 shows the same type of heatmap, but for building #2. Here, the strongest autocorrelation is also for 7 days back of 0.84. But with less correlation to one year ago (*lag1*). This can be due to the decreasing trend and change in energy usage pattern seen before. Building #2 does not have any values for 3 years back and therefore no values for *lag3*.

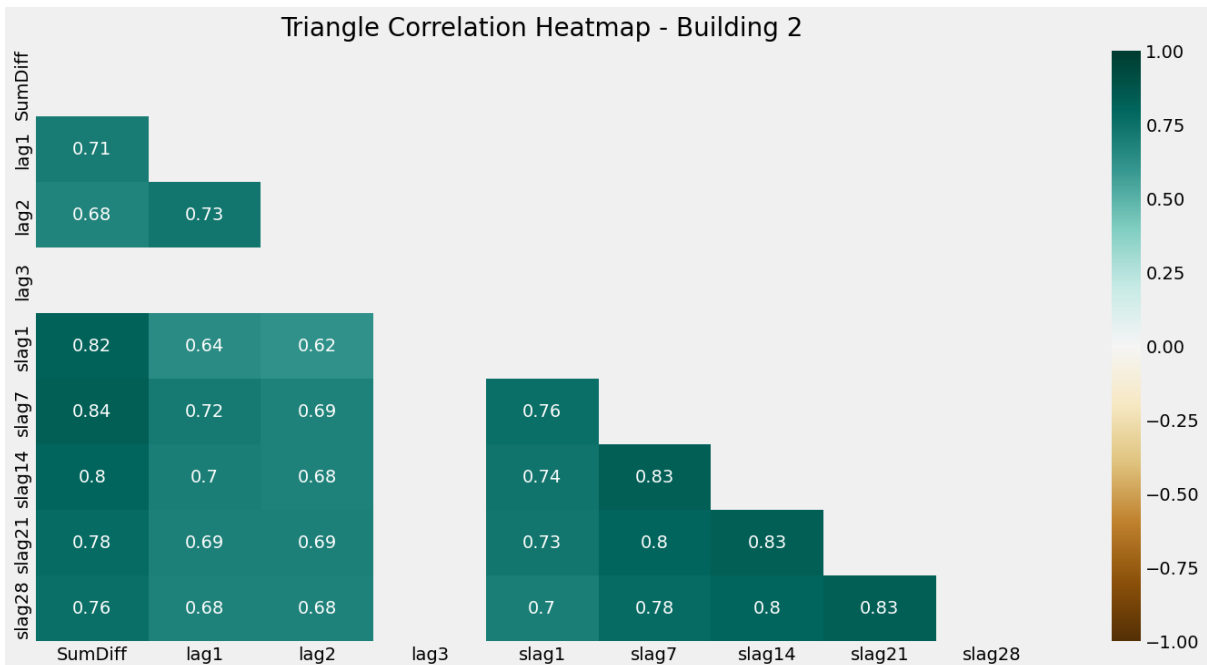


Figure 3-20: Correlation of lag features, building #2.

3.4 New temperature feature

Based on the previous scatter plots and correlation of important variables, it is seen that temperature is an important variable. Therefore, Figure 3-21 and Figure 3-22 takes a closer look at this relationship of temperature vs energy consumption, divided into two subplots: one for weekdays and one for weekends. Based on the previously presented hypothesis that 17 could be a base temperature for temperature dependent energy consumption. Here, a horizontal line for the mean value for temperatures below 17 (previously discussed *base temperature*) and a linear regression line for temperatures above 17 (red lines) are added.

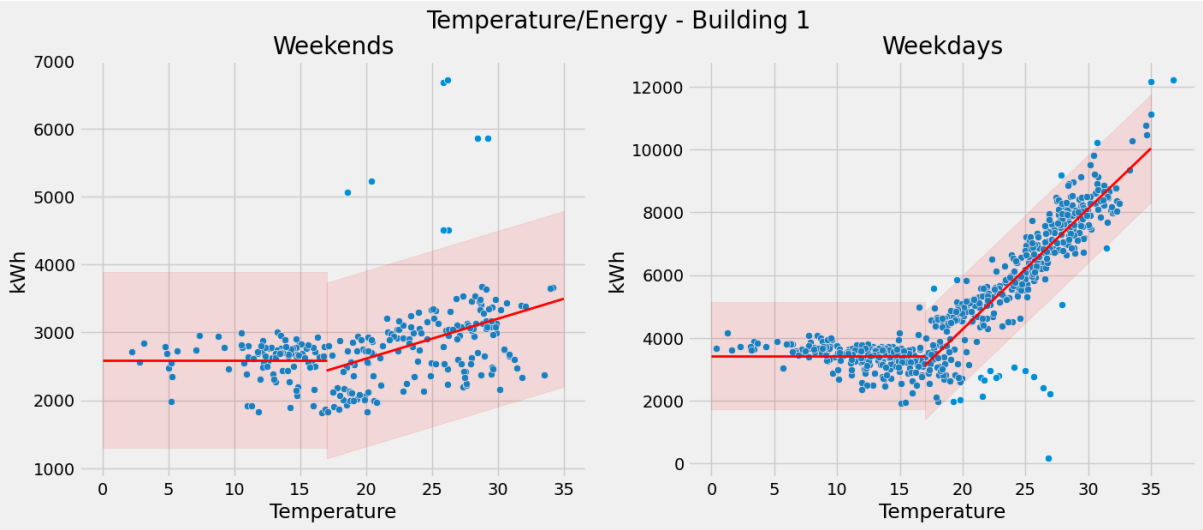


Figure 3-21: Scatter plot of temperature vs energy consumption building #1 grouped by weekday and weekend, aggregated for days. The red shaded areas are only a static distance from the red lines for easier visualization.

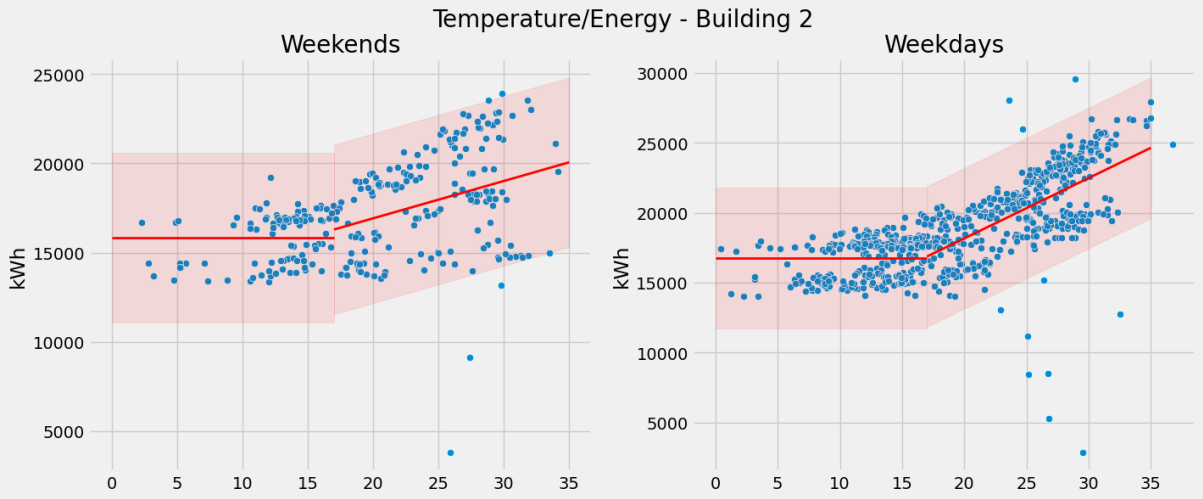


Figure 3-22: Scatter plot of temperature vs energy consumption building #2 grouped by weekday and weekend, aggregated for days. The red shaded areas are only a static distance from the red lines for easier visualization.

On the assumption that there might be a linear-like relationship between the energy consumption and the temperatures, when above 17 degrees and no significant tendency below (as seen above), a new variable *temperature_adjusted* is created. This new variable is then calculated by subtracting 17 to the temperature values and setting the resulting negative values to zero (temperatures below 17 => adjusted temperature = 0). This is another example of feature

3 Data analysis results

engineering done in this project. Figure 3-23 shows the new correlation heatmap with weather features including this adjusted temperature feature for building #1. From this figure, we can see that the correlation between energy consumption has increased from 0.62 for *temperature* to 0.69 for the *temperature_adjusted*. If we group samples into weekdays and weekends, the correlation is increased from 0.4 to 0.42 for weekends, and from 0.84 to 0.92 for weekdays. The same trend can also be seen for building #2, where general increased from 0.57 to 0.6. Weekends from 0.42 to 0.43, and weekdays 0.66 to 0.68.

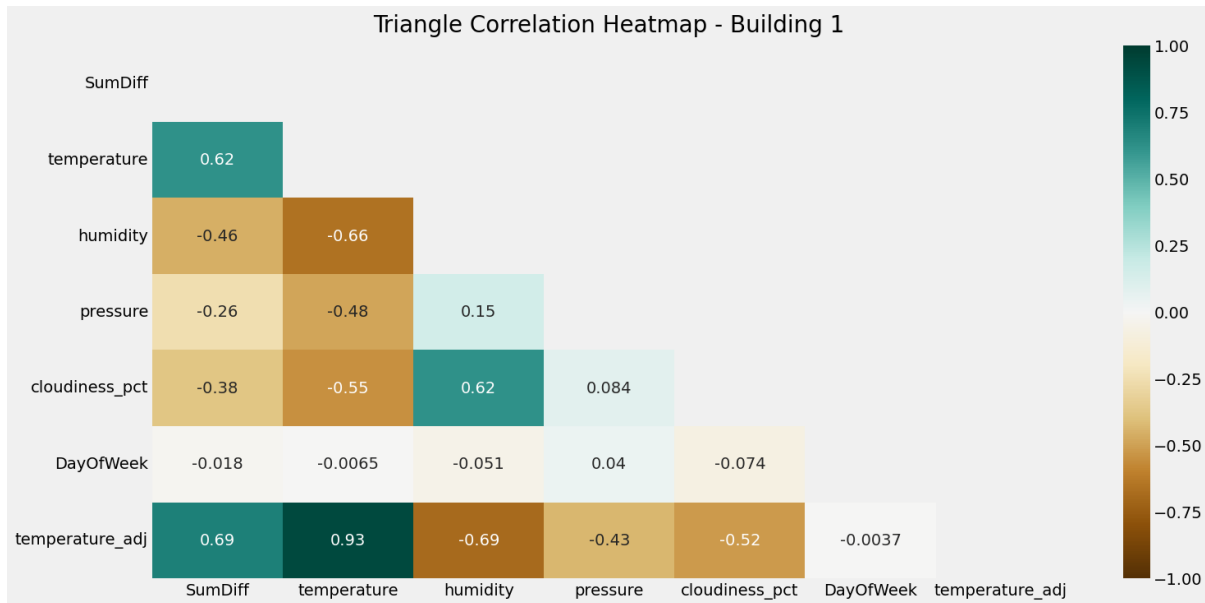


Figure 3-23: Correlation of weather features and building #1.

Figure 3-24 shows a scatter plot of the relation between energy consumption and this new adjusted temperature for building #1. In addition to the linear regression line in red, a second order regression line is added for visual effect as this seems to fit the values better, especially for the top end values in weekdays.

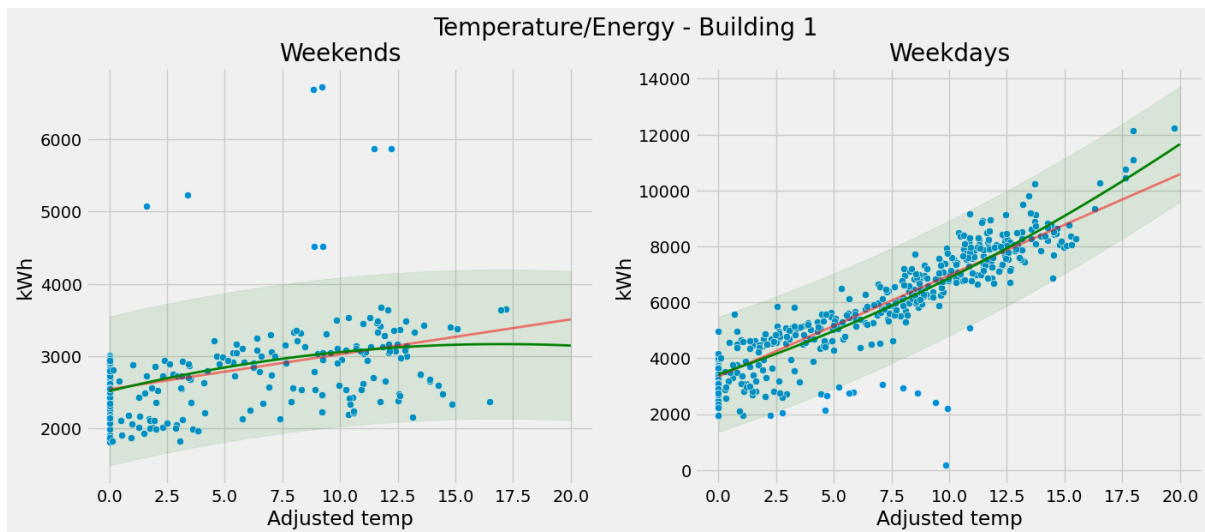


Figure 3-24: Scatter plot of energy consumption vs adjusted temperature for building #1. The green shaded areas are only a static distance from the green line for easier visualization.

4 Modelling results

This chapter will present the model results for a simple baseline model, XGBoost (extreme gradient boosting models) and LSTM models.

The available energy data is divided into training, validation, and test sets, where the splitting dates used are 2022.01.01 and 2022.08.01. This gives about 66% for the training set, and 17% each for the validation and test set. Figure 4-1 shows the split of available energy data for building #1.

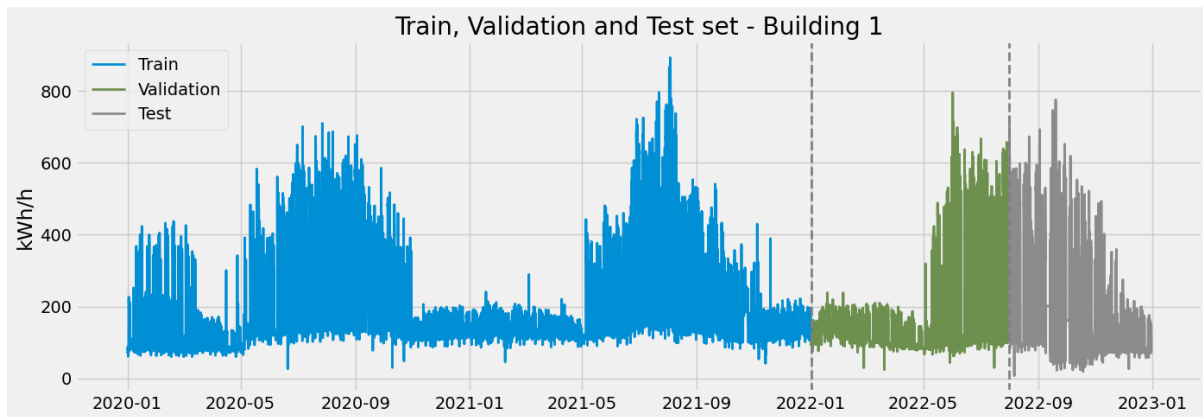


Figure 4-1: Train, validation, and test set split for building #1.

This split is used for the baseline model along with the gradient boosting models. The LSTM models will only have approximately this date split as they use a percentage of the total number of records, as opposed to the actual dates shown in the plot.

In the plots for model prediction output below, samples with high prediction error (RMSE > two standard deviations) are marked with a red circle in the model output plot to indicate areas with poor predictions compared with validation and test set.

4.1 Outlier handling

Since the data has both trend and seasonal components, simple statistical methods like IQR and standard deviations might be too simple approaches as the same values for one season might be an outlier in another season. Automatic outlier algorithms might also work, but oftentimes it is not known on what basis these select the outliers. Some methods like the isolation forest, the number of outliers in the data must be set in advance which makes it somewhat of a trial-and-error approach to finding the correct percentage of outliers present in the data.

Two custom methods for identifying potential outliers are developed and suggested as an alternative in this project. The first is Standard deviations From grouped Mean (SFM), the second is inspired by the previously created variable adjusted temperature.

4.1.1 Standard deviations From grouped Mean (SFM)

SFM is an algorithm for outlier detection developed during this project and is given the name SFM (Standard deviations From grouped Mean). It is a method that calculates a mean value

4 Modelling results

based on grouping by user-selected time features (ex. *DayOfWeek* and *Quarter*), and then identify values as outliers if the value is outside a given number of standard deviations from this grouped mean. It is a repetitive method that performs this operation in iteration, identifies and removes outliers in one round, then recalculates mean and standard deviation for next round with the previous outliers removed. Example output from this algorithm is shown in Figure 4-2, where outliers are marked with a red circle.



Figure 4-2: Output from SFM (Standard deviations From grouped Mean), building #2.

4.1.2 Temperature scatter

Another method for identifying potential outliers that is developed during this project is based on the previously described adjusted temperature (chapter 3.4). Values that deviate greatly from the rest of the temperature trend are marked as potential outliers. What boundary to use can be adjusted but a value of 0.3-0.5 times the mean is used here. Figure 4-3 shows an example from this method, where outliers are marked as red dots.

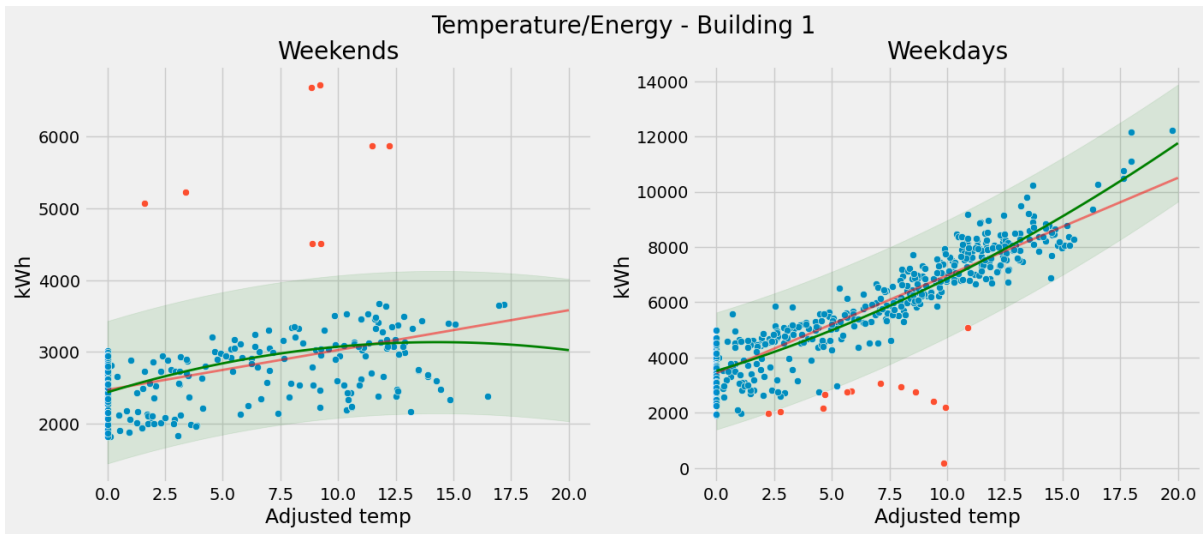


Figure 4-3: Marked potential outliers from temperature scatter, data from building #1.

4.2 Baseline model

A simple, common sense, no-learning model is created to set a prediction accuracy baseline value for what the machine learning models should beat in order of adding any value. From the variable analysis part, we saw that there is a strong autocorrelation between the current energy consumption and the previous year’s energy consumption (*lag1* feature). So, the simple baseline model used here is just to predict the future as a copy of the past. Then, the model’s forecast for the next period’s consumption will be equal to the previous time period’s consumption. In this model, the energy consumption from 364 days back (*lag1*) is used as the model output, but this could also have been done for a shorter prediction horizon by using the 7 days back and so forth. Figure 4-4 shows the output from this simple, no-learning model for predicting the daily energy consumption for the next 364 days in building #1.

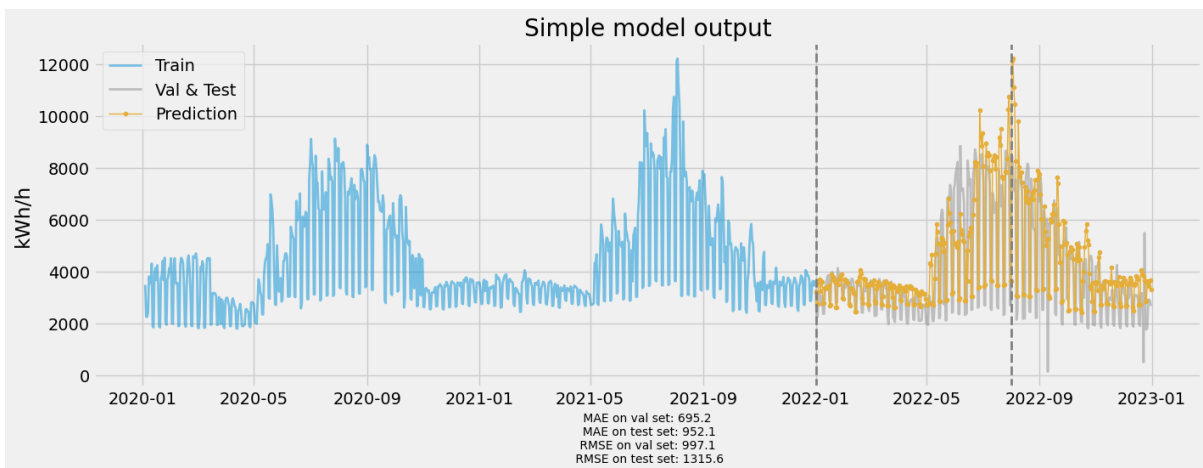


Figure 4-4: Predicting future as copy of last year, for building #1.

The full prediction results for this model type are shown in Table 4-1. This simple model can also include adjustment for the underlying trend in the data, which improves the predictions for building #2 but at the same time increases the error for building #1.

Table 4-1: Simple prediction model result

Dataset	Input Features	Model Parameters	Prediction horizon	MAE/RMSE Test set	
				Building#1	Building#2
Hourly	<i>Lag1</i>	Adjusted for trend	1 year	58/86	96/111
Hourly	<i>Lag1</i>	Not adjusted for trend	1 year	56/82	103/141
Daily	<i>Lag1</i>	Adjusted for trend	1 year	1054/1425	2495/3340
Daily	<i>Lag1</i>	Not adjusted for trend	1 year	997/1315	2666/3923

4.3 Extreme gradient boosting

This subchapter presents the result from XGBoost (extreme gradient boosting) models. Where the first model types predict the daily energy consumption by using only lag and timely features, the second also includes weather features. Lastly the models predict hourly energy consumption.

4.3.1 Timely features – daily energy consumption

Firstly, models including only lag and timely features are created. One key advantage with this approach is that these input variables are normally known for an extended time into the future making the maximum prediction horizon from the models further.

From the previous variable analysis, it is seen that the energy consumption has both daily and weekly cyclic behavior, and a seasonal variation during the year. Which also corresponds to the strong correlation between energy consumption and the lag features *lag1* and *slag7*. Therefore, the features: [*lag1*, *slag7*, *Week*, *Month*, *DayOfWeek*, *IsWeekend*] are believed to be of most importance and are the features in focus for these models. A script for automatically testing all combinations of these features is implemented to identify the most important features and combinations for the model. A result summary with the most important findings is shown in Table 4-2. These models use data aggregated for days and the prediction output will accordingly also be daily energy consumption. The full results from all testing all combinations can be found in Appendix B.

4 Modelling results

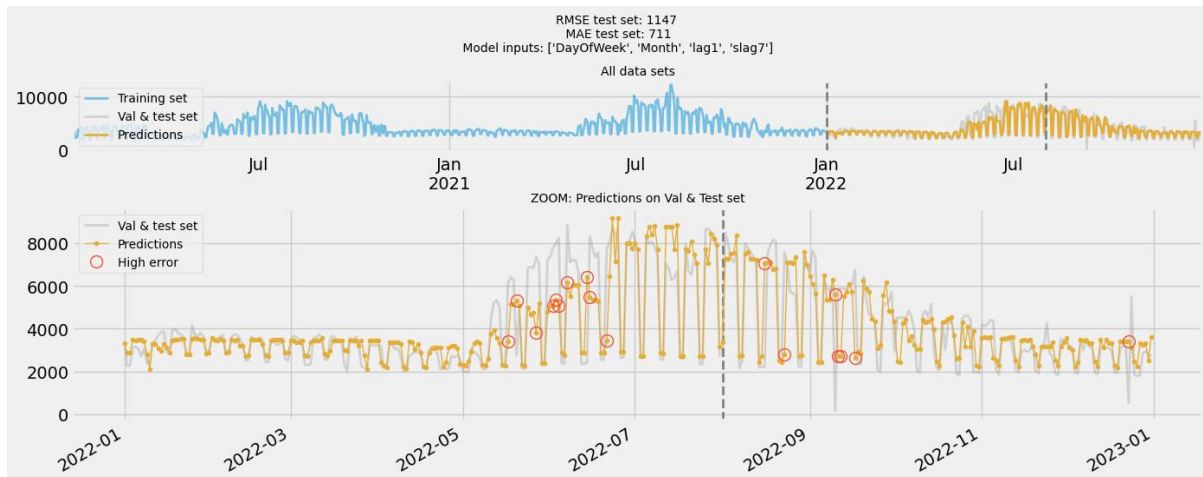


Figure 4-5: Simple model output building #1, aggregated for days.

Figure 4-5 and Figure 4-6 shows the prediction output from the best of these model types compared with the test set for building #1 and #2. The problem area for forecasting, in both buildings, seems to be in June (early summer).

Table 4-2: Model for daily consumption, without weather features. Summary, complete test result in appendix.

Dataset	Input Features	Maximum prediction horizon	MAE/RMSE Test set	
			B1	B2
Daily	<i>DayOfWeek, Month</i>	Unlimited	829/1168	1381/2762
Daily	<i>DayOfWeek, Week</i>	Unlimited	922/1234	1286/2759
Daily	<i>Lag1</i>	1 year	950/1239	1735/3083
Daily	<i>sLag7</i>	1 week	845/1394	1913/3274
Daily	<i>DayOfWeek, Lag1</i>	1 year	876/1150	1704/3074
Daily	<i>DayOfWeek, sLag7</i>	1 week	789/1322	1936/3296
Daily	<i>DayOfWeek, Month, lag1, slag7</i>	1 week	711/1147	1465/2792

4 Modelling results

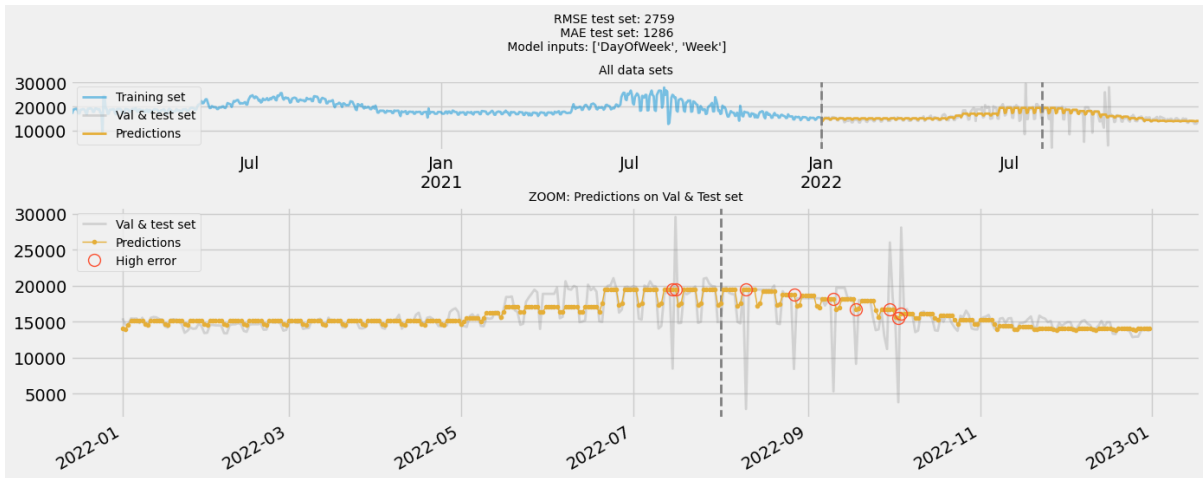


Figure 4-6: Simple model output building #2, aggregated for days.

4.3.2 All features – daily energy consumption

The parameters used in the previous model type are paired with the weather features showing highest correlation with energy consumption from the analysis part. These are: [*temperature*, *temperature_adjusted*, *cloudiness_pct*, *humidity*, *daylight*]. Figure 4-7 and Figure 4-8 shows the output from the models when also weather features are included compared with the test set for both buildings.

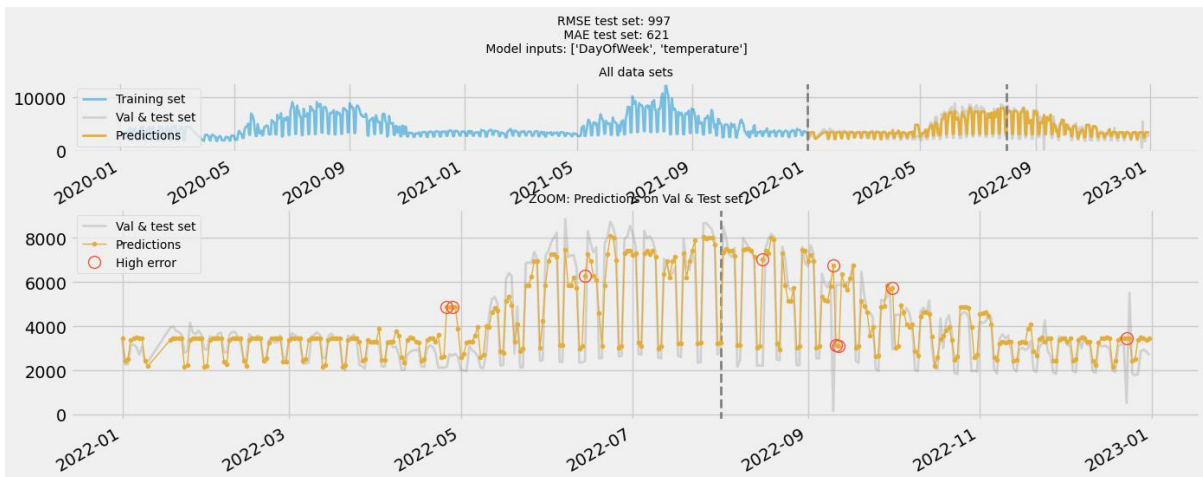


Figure 4-7: Model building #1 with the temperature parameter included, aggregated for days.

4 Modelling results

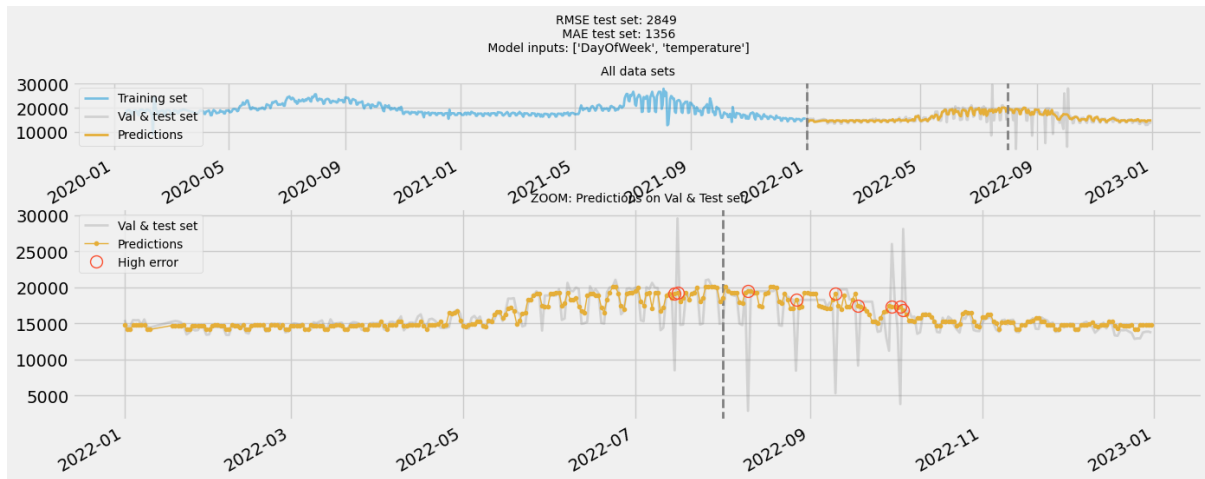


Figure 4-8: Model building #2 with the temperature parameter included, aggregated for days.

The script for automatically detecting the best combination of parameters is also used here, but the full test results are not included as an appendix because of the extensive length (10 variables gives over 1000 possible combinations). However, a summary of most important results for daily aggregated data can be seen in Table 4-3.

Table 4-3: Feature summary for models including weather features.

Dataset	Input Features	Maximum prediction horizon	MAE/RMSE Test set	
			B1	B2
Daily	<i>DayOfWeek, Temperature</i>	Forecast dependent	621/997	1356/2849
Daily	<i>DayOfWeek, temperature_adjusted</i>	Forecast dependent	612/990	1357/2839
Daily	<i>DayOfWeek, temperature, cloudiness_pct,</i>	Forecast dependent	631/1008	1345/2829
Daily	<i>DayOfWeek, Week, cloudiness_pct, daylight</i>	Forecast dependent	848/1205	1280/2702

4.3.3 All features – hourly energy consumption

Models for hourly energy consumption are also created. In addition to the features used in the daily models, the features [*Hour, IsWorkHour*] are also included. Summary of results for models using hourly data with and without the use of weather features are shown in Table 4-4.

Table 4-4: Short term model with and without weather data

Dataset	Input Features	Prediction horizon	MAE/RMSE	
			Test set	
			B1	B2
Hourly	<i>DayOfWeek, Hour, Month</i>	Infinite	50/72	51/68
Hourly	<i>DayOfWeek, Hour, Week</i>	Infinite	52/73	50/68
Hourly	<i>DayOfWeek, Hour, Temperature</i>	Forecast dependent	45/66	50/69
Hourly	<i>DayOfWeek, Hour, Temperature, IsWorkHour</i>	Forecast dependent	45/66	49/68
Hourly	<i>DayOfWeek, Hour, Temperature, IsWorkHour, slag7</i>	Forecast dependent, 1 week	41/70	53/76
Hourly	<i>DayOfWeek, Hour, Temperature, IsWorkHour, Week</i>	Forecast dependent	48/69	47/65

Figure 4-9 and Figure 4-10 shows the best model prediction output for the test set for both building #1 and #2 at an hourly interval.

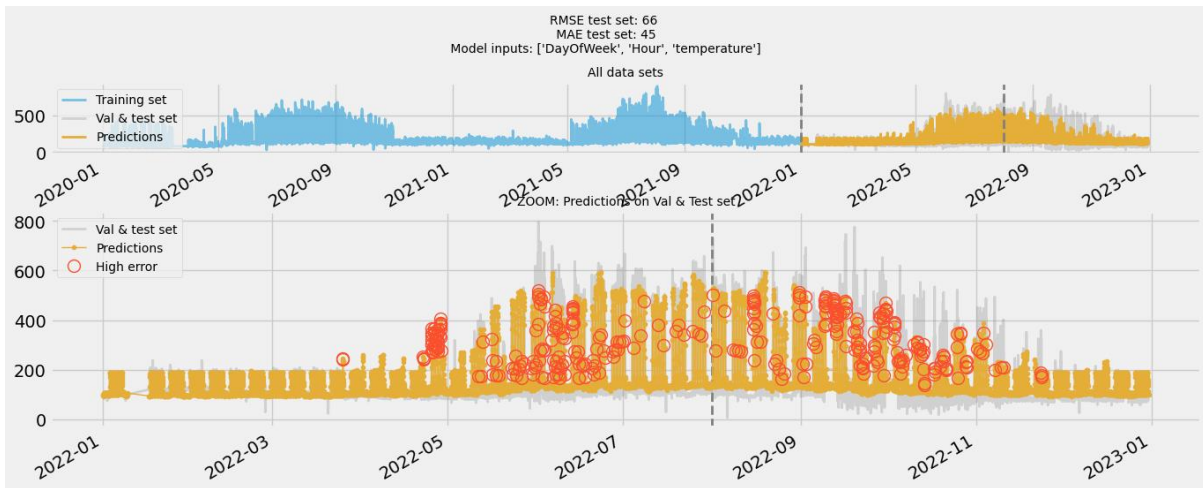


Figure 4-9: Hourly prediction with weather data included, building #1.

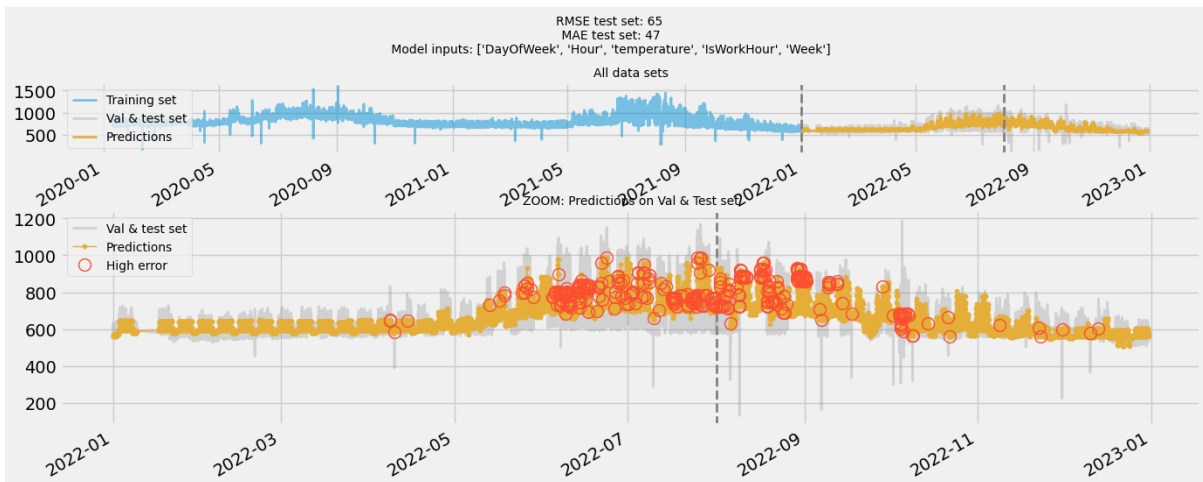


Figure 4-10: Hourly prediction with weather data included, building #2.

4.3.4 Model parameter importance

From the models created with the XGBoost library, it is possible to get the trained model's weighting of feature importance. It is seen that when both the features *DayOfWeek* and *IsWeekend* are included, the models prefer to emphasize only the *DayOfWeek* variable. The models using either one feature performs similarly, but slightly better with *DayOfWeek*.

Regarding the *temperature* and *temperature_adjusted* features, it seems that models including only the *temperature_adjusted* yields slightly better results. Although, when both features are included in the same model, the model emphasize only the *temperature* feature. Figure 4-11 shows a bar chart for feature importance from such a model.

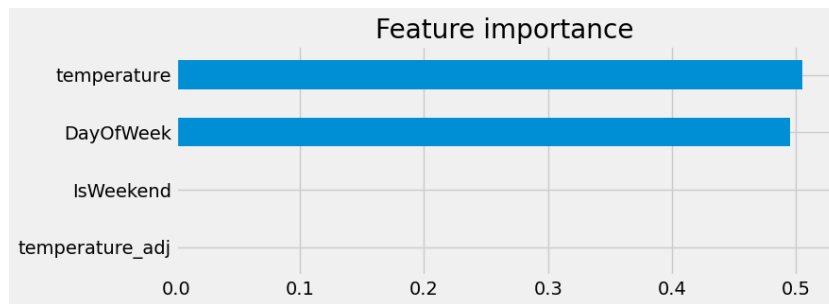


Figure 4-11: Feature importance, aggregated daily building #1.

When studying the features *Hour* and *IsWorkHour* in models where either feature is used, they seem to yield equivalent results. Although, the models inducing both parameters seem to yield a slight, but insignificant, improvement. Figure 4-12 shows a bar chart viewing the importance of each variable used by one of the hourly predicting models created with XGBoost.

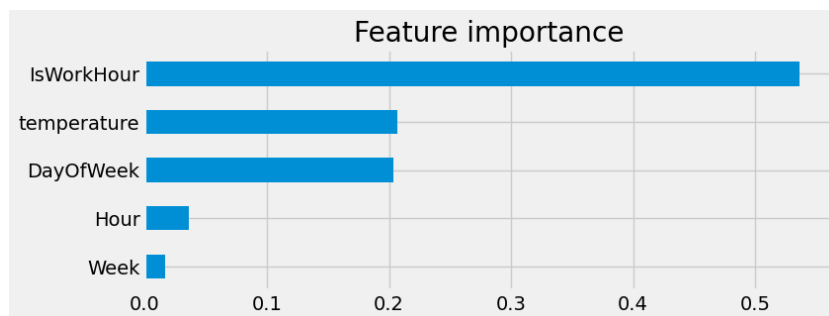


Figure 4-12: Bar chart over the importance of features in the model, building #1.

4.4 LSTM models

This subchapter presents the result from the LSTM models. The models presented below use a sequence of previous seven day's energy consumption as input to forecast the next day's energy consumption. It is also shown how a pre-trained model for one building can be adapted and re-trained for predictions on the other.

4.4.1 Single input model

The first model forecast the energy consumption for the next day given the energy consumption of the past 7 days - single input model. Table 4-5 shows the prediction results with different hyperparameter settings for building #1. Several MAE scores are included as training results for the same hyperparameter settings are different in each training session.

4 Modelling results

Table 4-5: Prediction results for LSTM simple model, building #1

Layer	Learning rate	Epochs before es	MAE Test set
1 layer: 7 node LSTM	0.1	11, 24, 57, 8, 35 epochs	1375, 897, 772, 1311, 849, 1317
1 layer: 14 node LSTM	0.1	56, 52, 31, 42	664, 714, 812, 792
1 layer: 32 node LSTM	0.1 (es =5 rounds)	25, 49	894, 1262, 697
	0.05 (es = 5 rounds)	54	668, 661, 690
	0.01 (es=10 rounds)	11,	1288, 768, 771
1 layer: 50 node LSTM	0.05 (es=10 rounds)	74, 42, 82	608 , 745, 643

Model prediction result on test set for the model with lowest MAE in building #1 value can be seen in Figure 4-13.

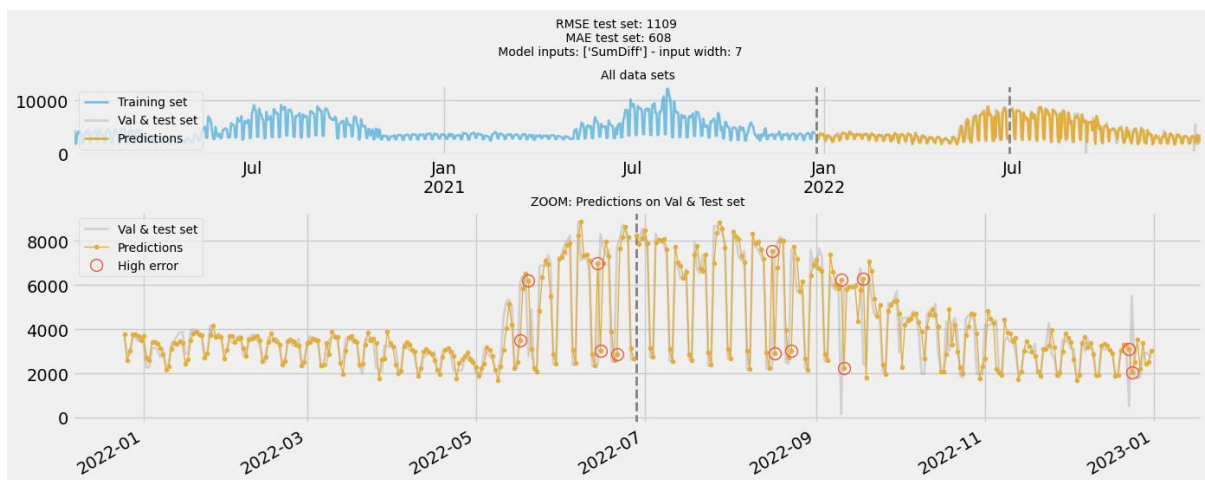


Figure 4-13: Predictions output from LSTM, building #1. One layer, fifty nodes

Figure 4-14 shows how training and validation loss function decreases with increasing epochs. Since the models use standardized values, the loss scale is in numbers of standard deviations.

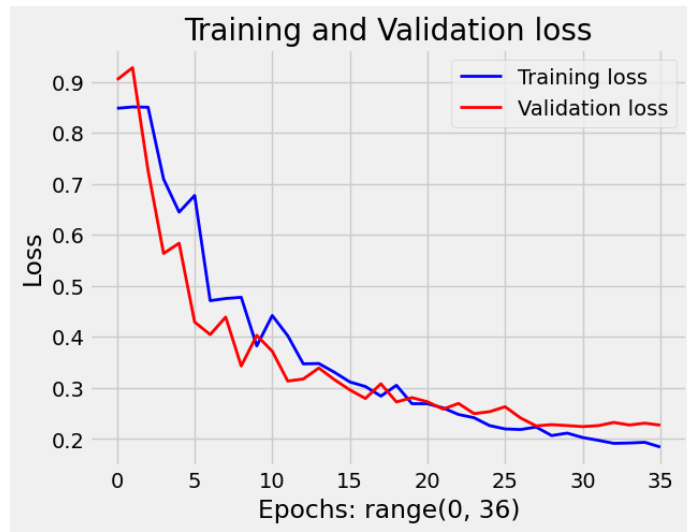


Figure 4-14: Loss for training and validation set.

4.4.2 General model

As seen in the data analysis part, the z-standardized values for both buildings have a strong pattern resemblance. So, by using the pre-trained model for building #1 (with 50 nodes as shown above) and simply just scale data (inputs/output) with standard deviation and mean corresponding to building #2 training set, the model also yields decent results as shown in Figure 4-15. This is then without any re-training of the model, just scale data to fit building #2.

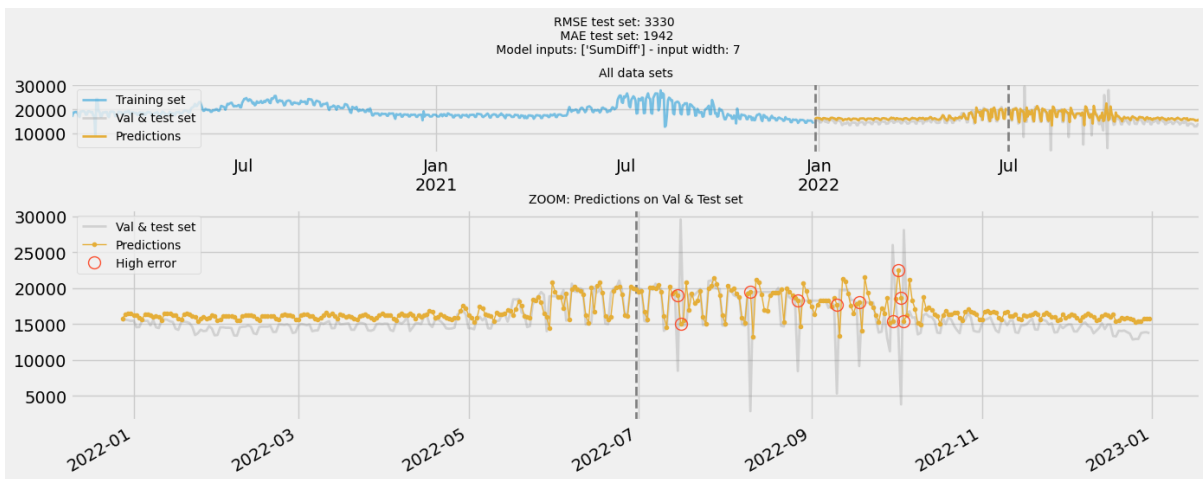


Figure 4-15: Model trained on building #1, just scaled to fit building #2.

However, the model’s performance is further increased when re-trained on building specific training data. Results after re-training shown in Figure 4-16.

4 Modelling results

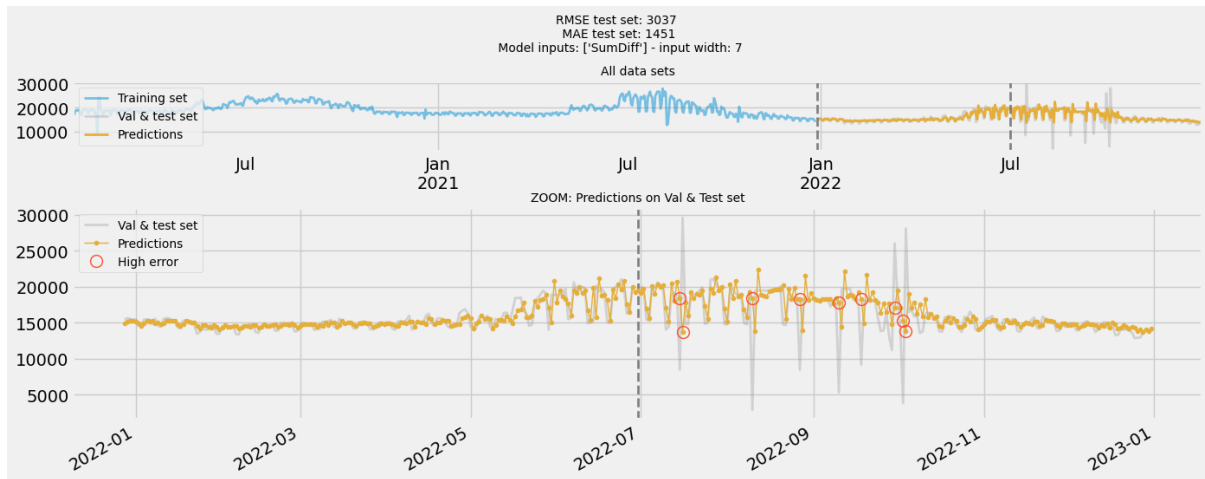


Figure 4-16: pre-trained model from building #1, retrained to building #2 data.

The inverse experiment where a model trained on building #2 data is scaled to fit building #1 is shown in Figure 4-17.

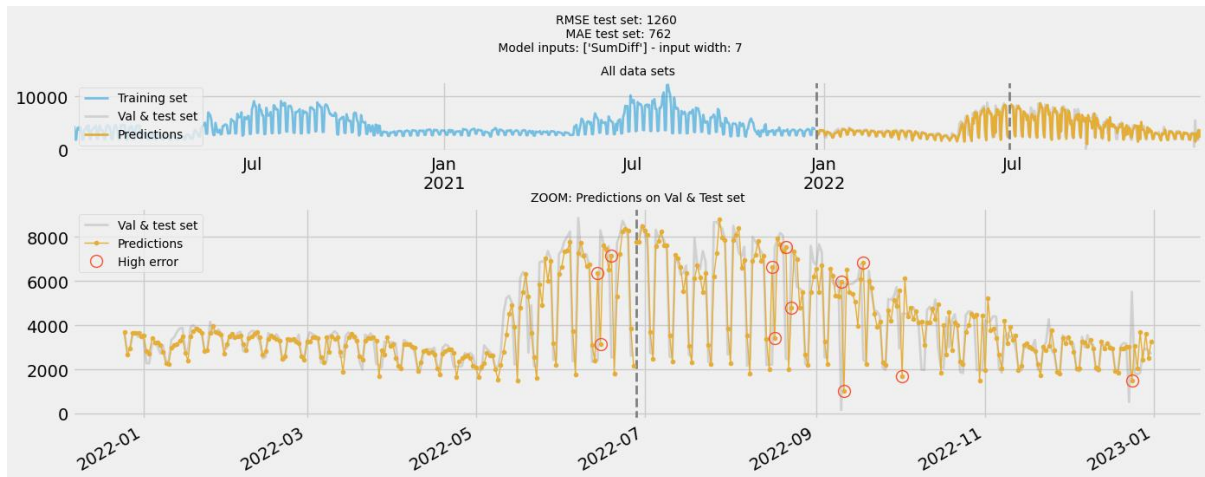


Figure 4-17: pre-trained model for building #2, just scaled to building #1.

4.5 Future predictions

For using any of the models shown above for future predictions (used in production), the best model settings can be used and retrained on all available data. The maximum prediction horizon will still be depending on the input variables used. Figure 4-18 shows an example of future predictions with an XGBoost model for building #1, where the maximum prediction horizon is one year according to model inputs.

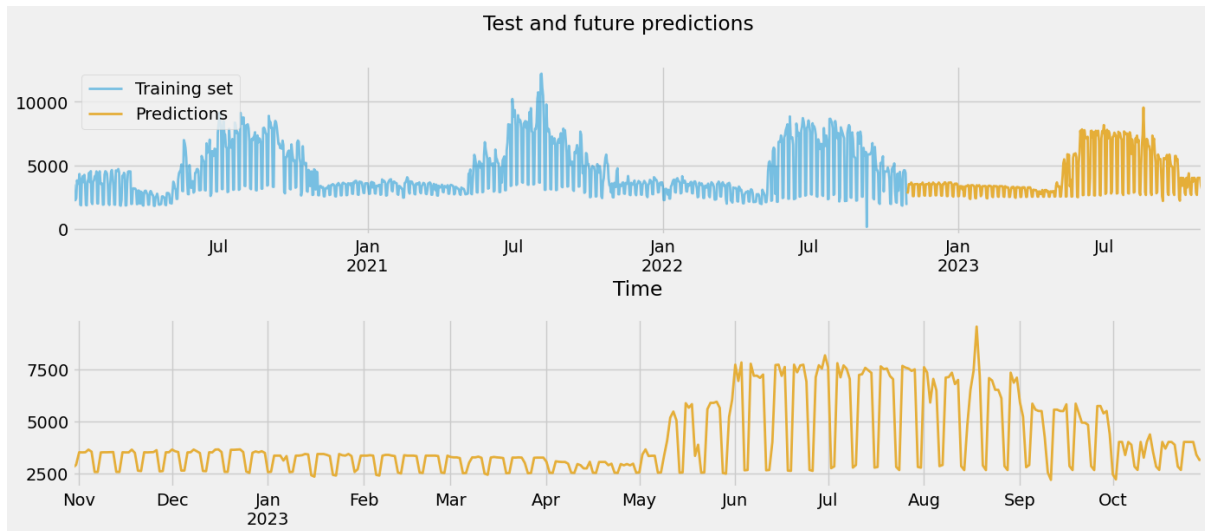


Figure 4-18: Model trained on all data for future predicting daily energy prediction, building #1.

5 Discussion

This chapter will discuss the findings, limitations and possible future work realized during the project. The chapter will begin with a discussion of the input data and the pre-processing, before moving to the data analysis and feature engineering. Then the models and the model results are discussed, and lastly future work.

5.1 Pre-processing

As earlier described, aggregated data is shifted forward in time during pre-processing. This will affect the timestamping of the data, and further the timely feature engineered variables like *DayOfWeek*. This shift in time can then create confusion during analyzing or visualizing the data. Example: Since events are shifted forwards, the events happening on a Sunday will be shifted forwards and marked as the following Monday (at 00:00). If data is then later grouped by day of week, it is data that are marked as Sunday and Monday that are the days belonging to weekend patterns (and not Saturday and Sunday). This is mostly an important concern when aggregation for days as the shift is larger but will also be the case for other aggregation intervals.

All data files contained some time slots with missing data, and it is selected to use linear interpolation for substituting missing energy data, and not to substitute for the missing weather data as these time slots were too large and therefore creates too big uncertainty. This will later affect how these variables can be used as inputs to models, and if the model type can handle missing inputs.

5.2 Data Analysis

The goal when developing any ML model is to make the model as simple as possible without sacrificing any of the prediction accuracy. To do so it is important to find the important parameters or features, which will provide the model with useful information and exclude noise. The data analysis part identifies that there are certain cyclical usage patterns for both buildings, as the consumption is lower on weekends compared with weekdays, higher during daytime compared with nighttime. Also, a seasonal pattern as consumption is higher during summer than winter indicating energy consumed for cooling. This is also confirmed with the correlation between the feature engineered lag factors, as there is a strong autocorrelation between now and the one week ago (*slag7*) and one year ago (*lag1 – 52 weeks*).

There is discovered a decline in the underlying trend for building #2, showing a reduction of about 20% energy consumption over the whole time period. It can also be signs that the usage pattern of building #2 has changed over the course of the three-year timespan, as it seems they have incorporated changes between workday/weekend, nightly reductions or similar. This may be seen in context with steep energy price increase in 2021 and even further in 2022, and consequently an overall increased attention to energy saving. It can be expected that this change will affect how well the models for building #2 can perform, as the pattern in training data will be changed during the timespan and not reflect the current situation that well.

When it comes to the weather features, it is identified that the outdoor temperature has the strongest positive correlation with the energy consumption in both building, and that

5 Discussion

consumption is highest during the summer period indicating cooling related energy consumption. Although building #1 has a larger percent wise dependency on temperature compared with building #2, when the data is standardized, the patterns are similar for both buildings. This fits well to the previously presented Enova report, where it's stated that the percentage of energy consumption that is temperature dependent will largely depend on the building type, but not least on what building materials or building standard is used. Enova has estimated that for commercial buildings the temperature dependent energy consumption can vary from about 25 to 70% depending on building material [6]. Even though the Enova report is referring to colder climates with heating of buildings, this seems to also cohere with cooling as seen in this report. So, this would suggest that the two buildings in this project are either used for different purposes, built after different building standards (age) or a combination of the two.

Of the remaining weather features, it is shown that cloudiness, number of hours with daylight and humidity correlated the most with the energy consumption. Many of these variables also had a strong correlation with temperature, indicating that they carry similar patterns and might not add any additional value to the models. If on the other hand some of these values have had a strong correlation with energy consumption, but not to temperature, one could expect that they carry relevant, new information to the models.

Beforehand, it might be intuitive to suggest that feels-like-temperature would be more important than solely temperature as this variable takes more information into consideration (an engineered feature). But the variable analysis shows that this is not true, at least for the two buildings in question during this project. This might be due to how the feels-like-temperature feature is calculated. Feels-like-temperature is often used as another name for apparent temperature, a combination of heat index and wind chill, which is a measure of the temperature perceived by humans. Where the wind chill is the cooling effect of wind to exposed skin, and heat index is a combination of temperature and humidity. The wind chill factor mostly influences temperatures below 10 °C and humidity factor for temperatures above 27 °C [26]. A range which covers the primary operating range for temperatures in this project. It is however reasonable to think that the feels-like-temperature variable could be more important when investigating heating of buildings in colder climates (below 10 °C), when this feature takes the wind speed factor into consideration.

In this case, it is expected that the sun will heat buildings and lead to an increase in the cooling demand and will consequently be an important factor in addition to just outdoor temperature. It could therefore be interesting to get a variable estimating the magnitude of this sun energy contribution for improving the energy models. More on this under section for future work.

5.2.1 Outlier detection

As earlier presented, there exist numerous strategies and algorithms for automatic outlier detection. The simpler methods are to remove values outside a certain standard deviation, or some inter quartile range (IQR). Because the data in this project varies much depending on time (season and cycles) these approaches were found to be a bit too simple whereas a value that is normal for one season can be out of range in another context. Also, some algorithms for automatic detection and removal of outliers are tried such as Local Outlier Factor, Isolation Forest, Elliptic Envelope and One class SVM (all available as libraries in scikit-learn). These

automatic outlier algorithms are just briefly tested but not used since it is hard to know exactly why they mark records as outliers.

Therefore, a couple of custom-made algorithms for detecting potential outliers are suggested in this project. Standard deviation From grouped Mean (SFM), and Temperature trend outliers. The key advantage with these approaches is that they are highly adapted to the specific purpose in detecting outliers in building energy data, and it is easy to understand the working principle. In addition to these two approaches, testing the effect of removing outliers from individual energy meters (before combining them) should be investigated since the energy consumption in each building is the sum of many energy meters within the same building. Here, the simpler, traditional outlier detection methods might be sufficient.

Also, all outlier methods used here only remove the outliers by dropping the sample. A better way could instead be to substitute the removed values, like the method used for substituting missing energy data during pre-processing.

5.3 Models

A simple non-learning model is created to set a baseline value for model accuracy. The simple baseline model is created on the assumption that the future energy consumption will be much like the past. For building #1 this simple model gave decent results, but for building #2 this method is not as satisfactory. This is probably because the underlying trend in the consumption is declining and the usage pattern is changed for building #2, but the predictions were somewhat improved when including the trend factor. This shows that this simple method can work to some extent in cases where the trend and usage pattern is stable, and likewise can get both better and worse by including the trend.

5.3.1 Model validation

For validating and measuring the model's performance, the metrics MAE (mean absolute error) and RMSE (root mean squared error) are used. Here, the RMSE values will penalize predictions with samples with large deviations harder than the MAE value, even if the number of samples with large deviation is small. In this project the MAE value is used to determine how well the models are performing, using RMSE instead could affect the results.

One tool that has shown to be effective to avoid overfitting and underfitting of models is the early stopping function. As earlier described, this function stops the model training when the predictions stop improving for the validation set. This enables setting the maximum number of epochs to a high enough number to prevent underfitting, and at the same time stops training before the model is overfitted. Special care is needed for finding the best suited early stopping number of rounds as this is closely related to the selected learning rate, particularly for LSTM training as the initial weights are random and training progress can be different each time.

5.3.2 Gradient Boosting Machines

It is shown that variations of XGBoost models can beat the simple baseline model's prediction accuracy for the test set. But the models for daily energy predictions with only lag and timely features as inputs had the largest prediction error area in the early parts of summer. This can

be due to a different climate for this period compared to earlier years and that this effect is not well reflected in the training data. Again, the model will only reflect the future as good as the training data.

This underestimation in early summer is improved by also including the outdoor temperature variable for building #1. Including temperature feature also made the *Week* and *Month* features obsolete and the best performing model for daily energy consumption in building #1 included only *DayOfWeek* and *Temperature*. For building #2 including the temperature feature did not enhance the prediction error in any significant way (test set MAE from 1286 to 1280). It is not concluded what is the exact cause for this, but it can be related to weaker relation to temperature compared to building #1 which percentwise has a higher temperature dependent energy consumption.

From the variable analysis it is shown that the created feature *adjusted temperature* has a greater correlation to energy consumption than *temperature*. Even though the adjusted temperature yielded higher correlation to energy consumption, the models preferred to use the standard *temperature* feature when both features are included as inputs, and the model predictions are disappointingly similar in both cases. This shows that XGBoost models can pick up on un-linear relationships in variables and adjust for them on its own. The same is also the case when using *DayOfWeek* and *IsWeekend* features in combination. The models prefer to use the variable with *DayOfWeek* in contrast to the one-hot encoded *IsWeekend*. It is expected that these one-hot encoded variables could be more important for the LSTM models.

The model's performance did not seem to have any significant improvements with the time features created from local time adjusted for DST, compared with UTC not adjusted for DST. This can be because the DST setting is mostly relevant to the Hour variable during summertime, which counts for an insignificant part of the model input importance.

One advantage with XGBoost compared to the LSTM method is that the training reaches the same result every time, so a set of variables or other settings only needs to be tested once. Also, that training is generally much faster. This enabled the possibility for creating a script for automatically testing all possible combinations of features in a convenient way.

5.3.3 LSTM

The LSTM model produced during this project was a model that only takes a sequence of the daily energy consumption for the previous seven days as input and predicts the next day's energy consumption. Some different network structures are tested, with the use of different numbers of nodes, learning rate and early stopping rounds. The results show that the best LSTM models can produce a lower prediction error value (MAE) for the test set compared with both the baseline model and the best XGBoost models.

Since the initial weights in a new model are random values, the output from a trained network will vary slightly each time it is trained. Since the training result can be different each time, it is important to train the models several times in order to establish if the settings are satisfactory. How good the trained models can be will also depend a lot on the selection of learning rate and settings for early stopping (hyperparameters). The effects of early stopping settings could perhaps have been mitigated if this setting had been switched off overall, and all the training reached the maximum number of epochs.

5 Discussion

Because the buildings showed such similar patterns when energy consumption was z-score standardized. It is shown that it is possible to use a pre-trained model for another building and just scale the data to the building specific standard deviation and mean. This could indicate the possibility to make kind of a general model that can be valid for multiple comparable buildings, which allows for making a decent model before much data from the specific building is collected. The pre-trained model is however improved through re-training with building specific data. Nevertheless, using a pre-trained model as the starting point for a new model training also helped the models reach a good solution faster, which then points to the benefit of having a good pre-trained model as the starting point whenever creating a model for a new building. This could for instance be accomplished by having a pre-trained model as a starting point for each comparable building type (school, office, apartment...).

The LSTM models mainly just improved the prediction accuracy for building #1, not for building #2. The test set for building #2 includes some heavy fluctuations in the energy consumption, indicating outliers or some other errors. Since the LSTM model will also use the test data as inputs, this will affect the model outputs in addition to the reduced MAE score on the output side. So, it might be a good idea to do some filtering on the model's input in order to isolate the effect from outliers to model output.

The LSTM models shown in this project use the last 7 days as input to predict the next day, but what if you like the model to predict the next 7 days rather than just the one day. There are mainly two ways of increasing the prediction width and thus getting multiple prediction horizons from the same model when using LSTM, one where the label width is increased during model training and then increasing the model's output. The other is making one day ahead prediction, as done in this project, but refeeding the output back to the model as an input making a multistep model.

5.3.4 Overall model results

In any case, the machine learning models did perform better than the simple, no-learning baseline model. Because of time limitations during the project, more time is put into the XGBoost models compared to the LSTM models, so it can be reasonable to think that the prediction accuracy for these models is closer to their maximum potential. It would therefore be interesting to further test how different model structures for the LSTM like deeper networks, different input width or more input variables could affect the results from LSTM models.

Furthermore, it is shown that the models were generally better at picking up the pattern at building #1 compared to building #2, even if the MAE score for building #2 has a percent wise better prediction accuracy. Again, this can be caused by the training data for building #2 not reflecting the current usage pattern of building #2 due to change in trend and usage pattern. Here, one possible solution could be to use just the most recent and relevant data during a re-training of the model and check if this can help performance.

During model training, the temperature measurement for that time slot is used, but when the model is to be used for future predictions (when in production), this measurement is obviously not available and a value from a weather forecasting service is needed instead. This raises the question of how this change will impact the model's prediction accuracy, which in turn depends on how closely related the forecasted temperature and actual temperature are. How far into the future we have a good temperature forecast, will also determine how far into the future these

model types can be used with a certain accuracy. In this project the assumption of up to seven days with a reliable temperature forecast is used.

One key advantage with the XGBoost models, is that the maximum prediction horizon is longer than with the LSTM and that we can create predictions for a year or more depending on the input variables selected. One of the problems with ML is the requirements for recorded data, therefore a more generic model type like shown with just scaling a pre-trained model here is desirable.

5.4 Future work

Some of the model predictions that are misplaced seem to be single days that are typically not fitting the “normal” pattern, further work could be investigated if these days are linked to national holidays or such. Including a variable indicating if a day is a national holiday is possible (ex. “*IsHoliday*”), but during a year it could be very few occurrences and thus little training data for the model, one simpler solution could then be to mark holidays as *IsWeekend* or set the day of week to Sunday, as it is reasonable to assume a similar pattern for Sundays and holidays.

Some features are only important when combined with others, and it is possible that these are not discovered in this project. As discussed above it would be attractive to get a variable describing the additional cooling demand caused by sun energy. Investigate the possibility for creating such a feature by combining daylight and cloudiness or check the resemblance to wet bulb temperature. Also, using more variables describing what happens inside the buildings could likewise help the model, like the indoor temperature measurement, occupancy level and so on. Here using data from a simulator could help identify which variables are important, before inserting new sensors and acquiring these from the actual buildings.

Because of time restrictions during the project, the handling of outliers and how different methods affect the model prediction result is not very well tested. A more in-depth investigation of outlier handling, substitution of removed outliers, input smoothing of outliers to LSTM inputs should be considered. Also possibly identify the explanation for outliers, some may be due to Covid-19 restrictions or national holiday.

On the model side further investigation of the LSTM networks, with other structures, deeper models, substituting LSTM with GRU nodes, more input variables should be investigated. Deeper and more complex models might call for the usage of dropout nodes during training. It is also possible to investigate other, new model types like Prophet. Since the building’s usage pattern often will change over time, the option for an automatic retraining, or automatic validation could be something to consider before setting the model into production.

The total energy consumption in each building is a combination of many energy meters within the building. Investigate if it is better to make models for each individual energy meter rather than the combined values. This way each energy consumer could use the model structure and input variables best suited for the specific purpose, like in [7].

6 Conclusion

This master thesis had two main goals: perform data analysis to identify important variables related to energy consumption and create machine learning algorithms for energy predictions in buildings based on this. Energy data for about a three-year period from two separate buildings located in Athens should be used as the foundation.

Before the data analysis could be performed a series of data pre-processing steps involving data aggregation, handling of missing values and feature engineering needed to be performed. Then, a comprehensive data analysis identified yearly seasonality, daily and weekly cyclical patterns for both buildings and underlying trend changes. Also, autocorrelation and correlation between energy consumption and other variables showed that outdoor temperature is one important variable, but it varies depending on the building and season (outdoor temperature).

To compare if the machine learning models provided any value, a simple, non-learning model was created in order of setting a baseline value for prediction accuracy. The baseline model is built on the assumption that the future will be like the past, and future predictions are therefore only a copy of the past. In any case, both machine learning types had a better prediction accuracy compared to this simple baseline model.

Many models and different configurations within two machine learning model types have been trained and tested during the project, these are LSTM and XGBoost models. The best performing XGBoost model for building #1 relied on *temperature_adjusted*, *DayOfWeek* as input variables, with the best results for daily energy predictions for test set MAE of 612 kWh (15% of test set mean value). And the best for building #2 used *DayOfWeek*, *Week*, *cloudiness_pct*, *daylight* as inputs with a MAE of 1280kWh (8% of test set mean value). The LSTM models used a sequence of the daily energy consumption from the previous seven days to predict the next day. Even with only a single input variable the best performing model had a MAE for test set of 608 kWh for building #1 and 1451 kWh for building #2.

Different buildings have different energy usage patterns, and the usage patterns can often change over time. It is hard to find one model type and one set of hyperparameters that will work well in all cases. The best performing model will therefore often depend on the specific building as seen in this project, where the best performing model for each building was of different type.

References

- [1] A. K. V.S.K.V Harish, "A review on modeling and simulation of building energy systes," *Elsevier*, 2015.
- [2] B. Marek and Z. Klaudia, "Prediction of Cooling Energy Consumption in Hotel Building Using Machine Learning Techniques," *MDPI energies*, 2020.
- [3] M. Elena , N. Phuong H., G. Madeleine and K. Wil L., "Comparison of Machine Learning Methods for Estimating Energy Consumption in Buildings," Eindhoven University of Technology, Department of Electircal Engineering, Eindhoven, The Netherlands.
- [4] M. Jin Woo , J. Sung Kwon, L. Yong Oh and C. Sangsun, "Prediction Performance of an Artificial Neural Network Model for the Amount of Cooling Energy Consumption in Hotel Rooms," *energies*, 2015.
- [5] A. S. C. Deb, "Review of data-driven energy modelling techniques for building retrofit," *Elsevier*, p. 13, 30 03 2021.
- [6] Enova, "Enovas byggstatistikk 2017," Enova , 2017.
- [7] F. J.-S. Abdul Afram, "Black-box modeling of residential HVAC system and comparison of gray-box and black-box modeling methods," *Elsevier*, p. 29, 03 11 2014.
- [8] F. Chollet, "Deep Learning with Python, second edition," Shelter Ilands, Manning Publication Co, 2021, pp. 1-67.
- [9] P. Kim, *MATLAB Deep Learning: With Machine Learning, Nerual Networks and Artificial Intelligenxce*, Apress, 2017.
- [10] J. Delua, "IBM, Supervised vs. Unsupervised Learning: What's the Difference?," IBM, 12 03 2021. [Online]. Available: <https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning>. [Accessed 20 03 2023].
- [11] S. Gupta, "Springboard - Regression vs Classification in Machine Learning: What's the Difference?," 6 10 2021. [Online]. Available: <https://www.springboard.com/blog/data-science/regression-vs-classification/>. [Accessed 16 04 2023].
- [12] F. Chollet, "Deep Learning with Python, second edition," Shelter Ilands, Manning Publication Co, 2021, pp. 280-308.
- [13] H. Viumdal, "Lecture notes: Introduction to Neural Networks," University of South-Eastern Norway, 2021.
- [14] R. Goyal, "Zeomag - Five Important Techniques That You Should Know About Deep Learning," 09 05 2018. [Online]. Available: <https://www.zeolearn.com/magazine/five-important-techniques-that-you-should-know-about-deep-learning>. [Accessed 20 03 2023].

References

- [15] M. West, "Bouvet - Explaining Recurrent Neural Networks," [Online]. Available: <https://www.bouvet.no/bouvet-deler/explaining-recurrent-neural-networks>. [Accessed 20 03 2023].
- [16] C. Olah, "Colah's blog - Understanding LSTM Networks," 27 08 2015. [Online]. Available: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>. [Accessed 25 03 2023].
- [17] IBM, "IBM - What are recurrent neural networks?," IBM, [Online]. Available: <https://www.ibm.com/topics/recurrent-neural-networks>. [Accessed 16 04 2023].
- [18] A. Biswal, "Simpli Learn - Recurrent Neural Network (RNN) Tutorial: Types, Examples, LSTM and More," 14 02 2023. [Online]. Available: <https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn>. [Accessed 25 03 2023].
- [19] "IBM - What are decision trees?," IBM, [Online]. Available: <https://www.ibm.com/in-en/topics/decision-trees>. [Accessed 28 03 2023].
- [20] "IBM - What is random forest?," IBM, [Online]. Available: <https://www.ibm.com/topics/random-forest>. [Accessed 28 03 2023].
- [21] IBM, "IBM - What is boosting?," IBM, [Online]. Available: <https://www.ibm.com/topics/boosting>. [Accessed 16 04 2023].
- [22] "Geeks For Geeks - Boosting in Machine Learning, Boosting and AdaBoost," 28 01 2022. [Online]. Available: <https://www.geeksforgeeks.org/boosting-in-machine-learning-boosting-and-adaboost/?ref=lbp>. [Accessed 28 03 2023].
- [23] T. Masui, "Medium - All you need to know about Gradient Boosting Machines - Part 1. Regression," 20 01 2022. [Online]. Available: <https://towardsdatascience.com/all-you-need-to-know-about-gradient-boosting-algorithm-part-1-regression-2520a34a502>. [Accessed 04 04 2023].
- [24] "XGBoost - Documentation," 2022. [Online]. Available: <https://xgboost.readthedocs.io/en/stable/index.html>. [Accessed 04 04 2023].
- [25] J. Bownlee, "Machine Learning Mastery - XGBoost for regression," 12 03 2021. [Online]. Available: <https://machinelearningmastery.com/xgboost-for-regression/>. [Accessed 04 04 2023].
- [26] "Pandas - About pandas," 2023. [Online]. Available: <https://pandas.pydata.org/about/index.html>. [Accessed 04 04 2023].
- [27] "Wikipedia, Apparent temperature," Wikipedia , 20 12 2022. [Online]. Available: https://en.wikipedia.org/wiki/Apparent_temperature. [Accessed 05 05 2023].
- [28] "Wikipedia, UTC," 02 03 2023. [Online]. Available: https://en.wikipedia.org/wiki/Coordinated_Universal_Time. [Accessed 13 03 2023].

References

- [29] "Wikipedia, Time zone," 01 03 2023. [Online]. Available: https://en.wikipedia.org/wiki/Time_zone. [Accessed 13 03 2023].
- [30] "Wikipedia, Dayligh saving time," 13 03 2023. [Online]. Available: https://en.wikipedia.org/wiki/Daylight_saving_time. [Accessed 13 03 2023].
- [31] W. Wheeler, "Medium," 27 05 2019. [Online]. Available: <https://medium.com/wwblog/clean-up-your-time-series-data-with-a-hampel-filter-58b0bb3ebb04>. [Accessed 05 04 2023].
- [32] "Enova - Enova kunnskap - Graddagstall," Enova, [Online]. Available: <https://www.enova.no/kunnskap/graddagstall/>. [Accessed 08 04 2023].
- [33] Dan, "<https://www.unixtimestamp.com/>," Dantools.com, 2014. [Online]. Available: <https://www.unixtimestamp.com/>. [Accessed 7 March 2023].
- [34] "Wikipedia, IOS 8601," 04 03 2023. [Online]. Available: https://en.wikipedia.org/wiki/ISO_8601. [Accessed 07 03 2023].

Appendices

Appendix A – Project topic description

Appendix B – XGBoost feature search results.

Appendix A – Project topic description



Faculty of Technology, Natural Sciences and Maritime Sciences, Campus Porsgrunn

FMH606 Master's Thesis

Title: Development of Predictive Machine Learning Algorithms for Energy Usage in Buildings

USN supervisor: Carlos F. Pfeiffer

External partner: Yodiwo (<https://www.yodiwo.com/>)

External Partner co-supervisor: Anthoula Mountzouri <an.mountzouri@yodiwo.com>

Task background:

While the importance of Energy Management has been growing for the last several years, it has now become a critical necessity. While the transition to renewable energy is still far from complete, the political situation of the world has restricted the access to traditional sources of energy, especially in Europe, anticipating a global energy crisis.

Machine Learning can provide Energy Management Systems (EMS) with diverse tools to register, forecast and optimize the use of energy in buildings and factories, and can help to mitigate the effects of the energy crisis.

Task description:

1. Literature research on Machine Learning techniques for prediction, with emphasis on applications to energy systems.
2. Analysis of existing data provided by Yodiwo for energy usage in buildings.
3. Selection and testing of different machine learning prediction models in buildings, using Python and data provided for Yodiwo.
4. Evaluation of the results.
5. Write and submit the Master Thesis report.

Student category: IIA

Is the task suitable for online students (not present at the campus)? Yes

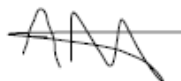
Practical arrangements: The thesis requires programming in Python, TensorFlow and Keras.

Supervision:

As a general rule, the student is entitled to 15-20 hours of supervision. This includes necessary time for the supervisor to prepare for supervision meetings (reading material to be discussed, etc).

Signatures:

Supervisor (date and signature): 30.01.23



Student (write clearly in all capitalized letters): ØYSTEIN GULDBERG

Student (date and signature): 30.01.23



Carsten F. Røhlfen

Appendix B – XGBoost feature search results.

Building #1:

combi	Features	Val_MAE	Val_RMSE	Test_MAE	Test_RMSE
61	['DayOfWeek', 'Month', 'IsWeekend', 'lag1', 'slag7']	607.0	1003.0	711.0	1147.0
50	['DayOfWeek', 'Month', 'lag1', 'slag7']	607.0	1003.0	711.0	1147.0
55	['Week', 'IsWeekend', 'lag1', 'slag7']	648.0	1089.0	756.0	1263.0
37	['Week', 'lag1', 'slag7']	652.0	1085.0	762.0	1270.0
62	['Week', 'Month', 'IsWeekend', 'lag1', 'slag7']	660.0	1084.0	764.0	1277.0
60	['DayOfWeek', 'Week', 'IsWeekend', 'lag1', 'slag7']	657.0	1074.0	773.0	1230.0
56	['Month', 'IsWeekend', 'lag1', 'slag7']	626.0	994.0	773.0	1197.0
47	['DayOfWeek', 'Week', 'lag1', 'slag7']	657.0	1074.0	773.0	1230.0
63	['DayOfWeek', 'Week', 'Month', 'IsWeekend', 'lag1', 'slag7']	648.0	1073.0	774.0	1232.0
59	['DayOfWeek', 'Week', 'Month', 'lag1', 'slag7']	648.0	1073.0	774.0	1232.0
48	['DayOfWeek', 'Month', 'IsWeekend', 'lag1']	662.0	967.0	775.0	1084.0
27	['DayOfWeek', 'Month', 'lag1']	662.0	967.0	775.0	1084.0
54	['Week', 'Month', 'lag1', 'slag7']	668.0	1078.0	781.0	1292.0
30	['DayOfWeek', 'IsWeekend', 'slag7']	563.0	959.0	789.0	1323.0
11	['DayOfWeek', 'slag7']	563.0	959.0	789.0	1323.0
49	['DayOfWeek', 'Month', 'IsWeekend', 'slag7']	562.0	937.0	791.0	1241.0
28	['DayOfWeek', 'Month', 'slag7']	562.0	937.0	791.0	1241.0
38	['Month', 'IsWeekend', 'lag1']	842.0	1142.0	797.0	1106.0
51	['DayOfWeek', 'IsWeekend', 'lag1', 'slag7']	607.0	956.0	799.0	1226.0
31	['DayOfWeek', 'lag1', 'slag7']	607.0	956.0	799.0	1226.0
40	['Month', 'lag1', 'slag7']	629.0	980.0	802.0	1214.0
17	['Month', 'lag1']	864.0	1158.0	803.0	1125.0
45	['DayOfWeek', 'Week', 'IsWeekend', 'lag1']	656.0	970.0	804.0	1124.0
39	['Month', 'IsWeekend', 'slag7']	574.0	975.0	804.0	1310.0
24	['DayOfWeek', 'Week', 'lag1']	656.0	970.0	804.0	1124.0
52	['Week', 'Month', 'IsWeekend', 'lag1']	821.0	1157.0	809.0	1108.0
57	['DayOfWeek', 'Week', 'Month', 'IsWeekend', 'lag1']	660.0	966.0	818.0	1134.0
43	['DayOfWeek', 'Week', 'Month', 'lag1']	660.0	966.0	818.0	1134.0
26	['DayOfWeek', 'Month', 'IsWeekend']	672.0	1013.0	829.0	1168.0
8	['DayOfWeek', 'Month']	672.0	1013.0	829.0	1168.0
46	['DayOfWeek', 'Week', 'IsWeekend', 'slag7']	600.0	1033.0	831.0	1235.0
35	['Week', 'IsWeekend', 'lag1']	821.0	1146.0	831.0	1129.0
25	['DayOfWeek', 'Week', 'slag7']	600.0	1033.0	831.0	1235.0
15	['Week', 'slag7']	626.0	1077.0	836.0	1330.0
18	['Month', 'slag7']	582.0	962.0	838.0	1337.0
6	['slag7']	604.0	1025.0	845.0	1394.0

Appendices

combi	Features	Val_MAE	Val_RMSE	Test_MAE	Test_RMSE
41	['IsWeekend', 'lag1', 'slag7']	624.0	969.0	847.0	1283.0
33	['Week', 'Month', 'lag1']	848.0	1176.0	847.0	1156.0
20	['IsWeekend', 'slag7']	598.0	996.0	848.0	1400.0
21	['lag1', 'slag7']	622.0	966.0	852.0	1291.0
14	['Week', 'lag1']	847.0	1171.0	862.0	1176.0
53	['Week', 'Month', 'IsWeekend', 'slag7']	629.0	1082.0	872.0	1346.0
36	['Week', 'IsWeekend', 'slag7']	641.0	1082.0	876.0	1334.0
29	['DayOfWeek', 'IsWeekend', 'lag1']	732.0	1044.0	876.0	1151.0
10	['DayOfWeek', 'lag1']	732.0	1044.0	876.0	1151.0
34	['Week', 'Month', 'slag7']	630.0	1082.0	886.0	1378.0
23	['DayOfWeek', 'Week', 'IsWeekend']	609.0	924.0	922.0	1234.0
7	['DayOfWeek', 'Week']	609.0	924.0	922.0	1234.0
42	['DayOfWeek', 'Week', 'Month', 'IsWeekend']	614.0	924.0	930.0	1260.0
22	['DayOfWeek', 'Week', 'Month']	614.0	924.0	930.0	1260.0
58	['DayOfWeek', 'Week', 'Month', 'IsWeekend', 'slag7']	602.0	1007.0	937.0	1321.0
44	['DayOfWeek', 'Week', 'Month', 'slag7']	602.0	1007.0	937.0	1321.0
5	['lag1']	766.0	1065.0	951.0	1239.0
19	['IsWeekend', 'lag1']	770.0	1070.0	956.0	1239.0
13	['Week', 'IsWeekend']	1063.0	1539.0	1113.0	1609.0
16	['Month', 'IsWeekend']	1130.0	1582.0	1120.0	1610.0
32	['Week', 'Month', 'IsWeekend']	1056.0	1533.0	1146.0	1640.0
12	['Week', 'Month']	1104.0	1592.0	1147.0	1674.0
2	['Week']	1105.0	1594.0	1151.0	1679.0
3	['Month']	1169.0	1646.0	1183.0	1659.0
9	['DayOfWeek', 'IsWeekend']	1438.0	1773.0	1310.0	1628.0
1	['DayOfWeek']	1438.0	1773.0	1310.0	1628.0
4	['IsWeekend']	1697.0	2037.0	1573.0	1878.0

Building #2:

combi	Features	Val_MAE	Val_RMSE	Test_MAE	Test_RMSE
23	['DayOfWeek', 'Week', 'IsWeekend']	902.0	1519.0	1286.0	2759.0
7	['DayOfWeek', 'Week']	902.0	1519.0	1286.0	2759.0
22	['DayOfWeek', 'Week', 'Month']	891.0	1515.0	1314.0	2761.0
42	['DayOfWeek', 'Week', 'Month', 'IsWeekend']	891.0	1515.0	1314.0	2761.0
13	['Week', 'IsWeekend']	1121.0	1780.0	1380.0	2795.0
8	['DayOfWeek', 'Month']	961.0	1560.0	1381.0	2762.0
26	['DayOfWeek', 'Month', 'IsWeekend']	961.0	1560.0	1381.0	2762.0
2	['Week']	1126.0	1783.0	1398.0	2820.0
32	['Week', 'Month', 'IsWeekend']	1118.0	1779.0	1427.0	2811.0
12	['Week', 'Month']	1122.0	1786.0	1461.0	2821.0
16	['Month', 'IsWeekend']	1164.0	1797.0	1512.0	2810.0
56	['Month', 'IsWeekend', 'lag1', 'slag7']	912.0	1491.0	1517.0	2911.0

Appendices

combi	Features	Val_MAE	Val_RMSE	Test_MAE	Test_RMSE
50	['DayOfWeek', 'Month', 'lag1', 'slag7']	889.0	1478.0	1518.0	2905.0
61	['DayOfWeek', 'Month', 'IsWeekend', 'lag1', 'slag7']	889.0	1478.0	1518.0	2905.0
40	['Month', 'lag1', 'slag7']	905.0	1489.0	1527.0	2914.0
3	['Month']	1170.0	1810.0	1530.0	2821.0
55	['Week', 'IsWeekend', 'lag1', 'slag7']	938.0	1524.0	1542.0	2898.0
37	['Week', 'lag1', 'slag7']	947.0	1525.0	1544.0	2906.0
31	['DayOfWeek', 'lag1', 'slag7']	832.0	1470.0	1548.0	2964.0
51	['DayOfWeek', 'IsWeekend', 'lag1', 'slag7']	832.0	1470.0	1548.0	2964.0
54	['Week', 'Month', 'lag1', 'slag7']	952.0	1525.0	1551.0	2899.0
41	['IsWeekend', 'lag1', 'slag7']	844.0	1474.0	1551.0	2963.0
62	['Week', 'Month', 'IsWeekend', 'lag1', 'slag7']	955.0	1524.0	1555.0	2901.0
21	['lag1', 'slag7']	849.0	1483.0	1559.0	2970.0
59	['DayOfWeek', 'Week', 'Month', 'lag1', 'slag7']	937.0	1514.0	1582.0	2926.0
63	['DayOfWeek', 'Week', 'Month', 'IsWeekend', 'lag1', 'slag7']	937.0	1514.0	1582.0	2926.0
60	['DayOfWeek', 'Week', 'IsWeekend', 'lag1', 'slag7']	959.0	1540.0	1587.0	2916.0
47	['DayOfWeek', 'Week', 'lag1', 'slag7']	959.0	1540.0	1587.0	2916.0
29	['DayOfWeek', 'IsWeekend', 'lag1']	1055.0	1666.0	1704.0	3072.0
10	['DayOfWeek', 'lag1']	1055.0	1666.0	1704.0	3072.0
36	['Week', 'IsWeekend', 'slag7']	1031.0	1642.0	1712.0	2947.0
58	['DayOfWeek', 'Week', 'Month', 'IsWeekend', 'slag7']	1009.0	1634.0	1724.0	2988.0
44	['DayOfWeek', 'Week', 'Month', 'slag7']	1009.0	1634.0	1724.0	2988.0
46	['DayOfWeek', 'Week', 'IsWeekend', 'slag7']	1012.0	1632.0	1725.0	2944.0
25	['DayOfWeek', 'Week', 'slag7']	1012.0	1632.0	1725.0	2944.0
19	['IsWeekend', 'lag1']	1081.0	1691.0	1726.0	3074.0
5	['lag1']	1089.0	1697.0	1735.0	3083.0
53	['Week', 'Month', 'IsWeekend', 'slag7']	1020.0	1646.0	1740.0	2975.0
15	['Week', 'slag7']	1032.0	1652.0	1741.0	2952.0
34	['Week', 'Month', 'slag7']	1021.0	1648.0	1762.0	2980.0
28	['DayOfWeek', 'Month', 'slag7']	1011.0	1638.0	1782.0	2977.0
49	['DayOfWeek', 'Month', 'IsWeekend', 'slag7']	1011.0	1638.0	1782.0	2977.0
18	['Month', 'slag7']	1043.0	1663.0	1805.0	2975.0
39	['Month', 'IsWeekend', 'slag7']	1035.0	1662.0	1807.0	2994.0
57	['DayOfWeek', 'Week', 'Month', 'IsWeekend', 'lag1']	898.0	1518.0	1849.0	3078.0
43	['DayOfWeek', 'Week', 'Month', 'lag1']	898.0	1518.0	1849.0	3078.0
24	['DayOfWeek', 'Week', 'lag1']	909.0	1527.0	1896.0	3100.0
45	['DayOfWeek', 'Week', 'IsWeekend', 'lag1']	909.0	1527.0	1896.0	3100.0
52	['Week', 'Month', 'IsWeekend', 'lag1']	995.0	1593.0	1912.0	3136.0
6	['slag7']	1082.0	1750.0	1913.0	3274.0
27	['DayOfWeek', 'Month', 'lag1']	898.0	1506.0	1928.0	3152.0

Appendices

combi	Features	Val_MAE	Val_RMSE	Test_MAE	Test_RMSE
48	['DayOfWeek', 'Month', 'IsWeekend', 'lag1']	898.0	1506.0	1928.0	3152.0
33	['Week', 'Month', 'lag1']	1002.0	1604.0	1933.0	3147.0
30	['DayOfWeek', 'IsWeekend', 'slag7']	1068.0	1743.0	1936.0	3296.0
11	['DayOfWeek', 'slag7']	1068.0	1743.0	1936.0	3296.0
20	['IsWeekend', 'slag7']	1074.0	1750.0	1938.0	3311.0
17	['Month', 'lag1']	964.0	1548.0	1940.0	3169.0
35	['Week', 'IsWeekend', 'lag1']	993.0	1596.0	1944.0	3149.0
14	['Week', 'lag1']	1002.0	1608.0	1958.0	3159.0
38	['Month', 'IsWeekend', 'lag1']	948.0	1534.0	1985.0	3224.0
9	['DayOfWeek', 'IsWeekend']	1710.0	2187.0	2127.0	2975.0
1	['DayOfWeek']	1710.0	2187.0	2127.0	2975.0
4	['IsWeekend']	1838.0	2340.0	2191.0	3032.0