university of groningen

faculty of mathematics and natural sciences

Discovering Features for a Smart Heating System

Master Thesis Computing Science

Wednesday 31st August, 2016

Student: Bas de Bruijn

Primary supervisor: Prof. dr Alexander Lazovik

Secondary supervisor: Dr. Mircea Lungu

Daily supervisor: Dr. Ilche Georgievski

Contents

Abstract

When modeling data, the selection of features is important in determining the quality of the model. For this thesis we explore the possibility of discovering new features by clustering the data in different ways. We assess these feature engineering techniques in the context of a smart heating system. A smart heating solution is designed, implemented and deployed in an office building with about 100 offices. Having an accurate model of the office temperature is important when trying to save energy, as the model allows us to predict when we can turn down the heating system. Thus, we assess whether or not we can improve the models we have using these feature engineering techniques. The method of choice for this assessment is a form of cross-validation, where we leave one of the features out from the model creation, and see if we can derive its existence from the data.

Chapter 1

Introduction

Data is playing a more prominent role in most of our lives everyday. As sensor technology advances we are able to measure more aspects of the world and as a result produce more information about it. Combined with the increase of more abstract data such as server uptime or number the of financial transactions, the amount of data that is being produced is increasing at a fast rate. One way to make sense of this massive amount of data is by creating a model of the data. A model functions as a description of the data, that can usually be used to infer unknown values at a later point. Figure 1.1 illustrates this in the case of a simple linear regression model. The blue dots are two-dimensional data points, and the red line running through it is a model of the data. This red line has the advantage that it can be represented using much less storage than the original data points. In this scenario one can use the model to determine an estimation of the y-value given some value of x.

This ability to estimate data points using a model is very useful in many cases. Think for example of predicting the electricity demand in a smart grid system, or predicting the course of financial stocks. Different methods of modeling data exist, and they will use the available information in different ways to obtain an estimation of whatever it is that needs to be estimated. For those models that use *features*, variables that are used to predict some other variable, the choice of features determines for a large part the quality of the model. In the case of Figure 1.1 the only feature is x, which is used to predict y. When y is more concrete like, for example, the electricity demand in a smart grid system, the choice of features has to be appropriate for that variable. This topic is the focus of this thesis, because while some variables are quite obviously part of a model, more subtle variables may not be. Thus, in this work we propose and evaluate a method for discovering such features, a process known as feature engineering.

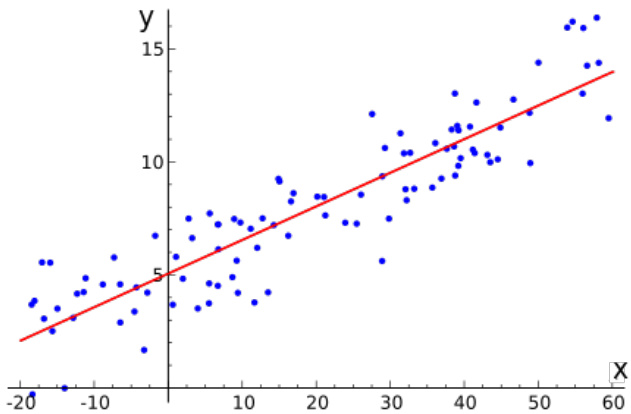


Figure 1.1: Linear regression line running through 2-dimensional data points [16].

Aside from the feature engineering method we also design and implement a smart heating solution. The feature engineering method serves to improve the quality of the smart heating solution. At the same time, the smart heating solution is used in this thesis to assess how well the feature engineering method works. The smart heating solution serves another purpose besides this, as the productivity and well-being of people working in an office environment is directly affected by the temperature of the office [37, 47]. Participants of the study performed by Lan et al. [37] showed lower motivation to do work in a moderately uncomfortable environment. Warm discomfort negatively affected participants' well-being, and they had to exert more effort to maintain their performance in an uncomfortable environment as opposed to a comfortable one. The study by Seppänen et al. [47] showed the highest productivity to be at around 22°C, while a temperature of 30°C resulted in a performance reduction of 8.9%. Maintaining the right office temperature is clearly important to the occupants and the businesses they work for.

When thinking about how to control the heating system one can identify two opposing forces. On the one hand we want to minimize the time that the occupant is in an uncomfortable environment. This means that the room should not get cold enough to be experienced as uncomfortable, i.e. a minimum temperature should be reached. On the other hand we would like to minimize the amount of energy used to heat a building. Saving energy is not only beneficial for those paying the energy bill, it also has a positive impact on the environment. This trade-off between maintaining the comfort of occupants while trying to save as much energy as possible is one that needs to be solved in order to create a sustainable heating solution. The fact that the temperature of an office is important is evident, and this is also the reason that this is the variable that we want to predict.

These problems lead to the following set of research questions that this thesis aims to answer:

1. How can we discover new features that can be used for improving data

models?

2. How can context be derived from a smart heating system?
3. How can the negative effect of a smart heating system be minimized, based on the gathered context information? That is to say, how can we control the radiators in such a way that we optimize the trade-off between user satisfaction and saving energy?

The remainder of this thesis contains the following. In Chapter 2 we look at the related work on these two topics: feature engineering and smart heating. Chapter 3 presents the design of both the feature engineering method and the smart heating solution Because the smart heating solution has been deployed in a reasonably large building, we go over the deployment details in Chapter 4, as well as the implementation details. The project is evaluated in Chapter 5, after which the thesis concludes with Chapter 6, conclusion and future work.

Chapter 2

Related Work and Background

This thesis covers two major topics, the first of which is about discovering new features for linear regression models. The second topic is about the realization of a smart heating system. They are related to each other in the sense that the former is intended to function as a set of tools to improve the latter. Both topics come with some areas for which presenting some related work is warranted. We focus on four main topics:

1. Feature creation
2. Smart heating systems
3. Modeling room temperatures
4. Clustering time series

2.1 Discovering Features for Regression Models

This section looks at two related papers on creating features. Both studies apply feature selection rather than feature creation. These studies are interesting because they approach the problem of arriving at some ideal set of features from a different angle than this thesis. Rather than trying to come up with new features based off the data, they have a large set of features, many of which irrelevant, and then attempt to reduce this large set of features to the ideal set of features.

2.1.1 An introduction to variable and feature selection - Guyon and Elisseeff

Guyon and Elisseeff [30] present an introduction to variable and feature selection. They define the difference between a variable and a feature as: "We call

"variables" the "raw" input variables and "features" variables constructed for the input variables." This distinction is not relevant for our purposes, as it is used to accommodate a different use case in the paper separate to ours. This paper presents works that approach the problem of feature selection from another angle than we do. They mainly deal with *dimensionality reduction*, i.e., they already have a lot of features, so many in fact that they are looking to reduce the number of features. Where we attempt to think of new features in a top-down manner, the work presented in [30] works from the bottom-up. This is an interesting approach in that one could in theory create a model from a lot of different features that do not even have to make sense, and then try to reduce these features in a way that is more sensible. Of course, this approach is more applicable in some cases than in others. Sometimes there are naturally many features available, while other times it will be difficult to find many features.

The paper presents multiple ways in which this reduction of features is achieved. The first is by means of clustering, features that are related to each other, i.e., that belong to the same cluster, are reduced to one feature: the cluster center. The most popular algorithms for this are k-means and hierarchical clustering. Clustering is coincidentally also one of the techniques we use, but with a different application. The other two techniques that are used are Matrix Factorization and Supervised feature selection.

2.1.2 A review of feature selection techniques in bioinformatics - Saeys et al.

Saeys et al. [45] present an overview of feature selection techniques, like [30] with a bottom-up approach. Feature selection techniques select a subset of the features rather than adjust the original representation of the variables, as is the case for other dimensionality reduction techniques such as ones based on projection or compression. The advantage of not adjusting the original variables is that they remain interpretable by domain experts. The goal is to end up with a minimal set of features, so as to minimize the risk of overfitting, provide faster and more cost-effective models and potentially gain deeper insight into the underlying processes that generated the data. Three categories of feature selection techniques are identified: filter, wrapper and embedding techniques. For the filtering techniques features are assigned a relevance score, and features with a low relevance are removed. Embedded techniques generate various subsets of features and then evaluate them. The embedded class of techniques combines the search for an optimal subset of features with the classifier construction.

2.2 Room Temperature Modeling

Having a model of the room temperature inside a building gives way to predicting the temperature in the future. Having an accurate prediction is important if the heating system is to be controlled optimally. In this section we review some existing works on modeling the room temperature.

2.2.1 A Physical Model for Estimating Air Temperature in Heating Systems - Liao and Dexter

Liao and Dexter [40] propose a method to model the air temperature in a building using physical parameters. Assessing a system that works with boilers and radiators, they use various physical properties of the building in order to estimate the air temperature. Figure 2.1 shows some of these parameters, such as radiator radiation, solar radiation entering through the window, heat infiltration through the window and conduction of the walls. They define a number of equations that are used in this modeling, relying on up to ten parameters to be known: from heating capacity of the boiler to the inertia of the water system to a number of heat transfer coefficients.

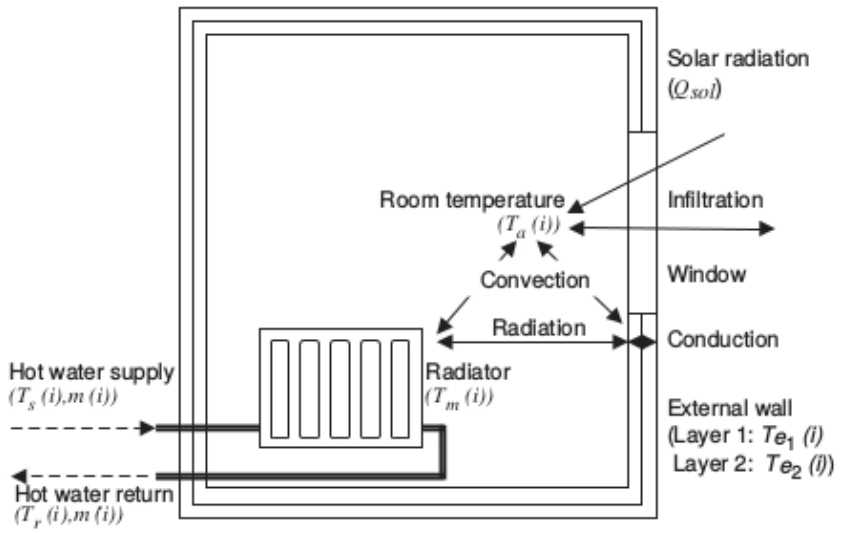


Figure 2.1: Heat transfer in one zone in a multi-zone heating system. Image and caption adapted from [40].

While it results in a good model of the air temperature, it requires a tremendous amount of expert knowledge about the building infrastructure. If such an extensive physical model is to be created every time one wants to equip a building with a smart heating system, this would be much more expensive than using a model that does not require this detailed information. It is for this reason that we choose not to go for this approach. Ideally we would want to have a model of the room temperature with as little required knowledge as possible. This is mostly a practical motivation, it is simply not desirable to invest so many resources into modeling the room temperature when dealing with a large number of buildings. This latter approach can be seen as a data science approach, where one tries to work with the data that is available in order to achieve some goal: in this case modeling the room temperature. Seeing as this data is already available, the required resources compared to the physical modeling approach are far fewer.

2.2.2 Modeling Temperature Using Autoregressive Models - Rios-Morena et al.

Ríos-Moreno et al. [44] compare two methods for modeling the room temperature: autoregressive with external input (ARX) and autoregressive moving-average with external input (ARMAX). These methods work similarly to regular AR or ARMA methods, with the exception that they contain a term for exogenous variables. These are variables that are determined by factors outside of the model. The external variables that are used for the prediction are:

- Outside air temperature
- Global solar radiation flux
- Wind speed
- Outside air relative humidity

with the inside temperature being the output variable. This work differs from our work in that they use the last few measurements to guide the next prediction. This has proven to work well, however, since we have to control the actuator fairly far in advance due to its delayed effect, we choose to explore a different method for modeling the data.

2.3 Smart Heating Solutions

Several case studies exist that implement some form of a smart heating solution. In this section we look at three different solutions.

2.3.1 PreHeat - Scott et al.

Scott et al. [46] deployed a system to control heating in homes, called PreHeat. It uses occupancy sensing and occupancy prediction to enable more efficient heating than a regular heating system. It has been deployed in five homes: three in the United States and two in the United Kingdom. In the UK homes the temperature can be controlled on a per-room basis, while the homes in the US were controlled on a per-house basis. The hardware was constructed by the researchers, and consisted of temperature sensors, motion sensors, control units and RFID receivers and tags, among others. Three different algorithms were used to determine the temperature set point:

Scheduled This algorithm acts like a programmable thermostat, where start and end times of presence are preconfigured. A distinction is made between being away, being present, and being asleep. The sleep time was also preset.

AlwaysOn The temperature is kept at the same set point for all presence states.

PreHeat The proposed prediction algorithm based on current and predicted occupancy. Heating control is realized by looking ahead for three hours and determining what the set point should be based on the presence prediction. Using the heat rate, i.e. the rate at which a room can be heated, the system determines whether or not the heating should be turned on or off.

The presence prediction of PreHeat works in two ways. First it reacts to the actual presence state as detected by the sensors. Second, it predicts future presence by discretizing presence into a binary vector, where each element represents the presence in some time interval using a boolean value. The prediction is then conducted by matching the presence vector of the current day so far to historical presence vectors of previous days. The Hamming distance is used as a similarity metric. The K most similar days are then used to predict occupancy, by computing the mean presence for each required future time interval.

PreHeat was evaluated with two measures: the measured gas consumption and the MissTime [41]. MissTime is defined as the total time that the home was occupied but not heated to the temperature set point. In the two UK houses the PreHeat algorithm performed better than the Scheduled algorithm on both metrics. Gas usage was reduced by 18% and 8%, respectively. MissTime was reduced by 38% and 60% respectively. MissTime in the US houses saw a large reduction: 84%, 88% and 92%. Gas usage for the US houses was similar to what it was before. Predictive heating plays a very significant part in reducing MissTime. This study presented some of the fundamental concepts on which our solution is built. Their control algorithm focuses on the presence prediction, while our control solution focuses more on the prediction of the room temperature using a number of different predictors, or features. It also differs in its application, we are more concerned with an office setting, while PreHeat is aimed at a domestic environment.

2.3.2 Smart Thermostat - Lu et al.

Lu et al. [41] propose a smart heating solution called "smart thermostat", operating with an HVAC system (Heating, Ventilation and Control). The environment is sensed using inexpensive wireless motion and door sensors. These sensors are used to infer whether occupants are away, active or asleep. Two main challenges are identified: 1) quickly and reliably determining when occupants leave the home or go to sleep, and 2) deciding when to turn on the HVAC system. HVAC systems often have multi-stage heating components, with one very efficient slow-heating component, and higher-powered fast-heating component. A heating system that would be reactive, i.e. turn on the heating once arrival has been detected, would actually use more energy as the high-powered component would do a lot of work. The approach is evaluated in 8 homes.

The smart thermostat employs three different energy saving techniques. The *fast reaction* algorithm determines whether occupants are away, active or asleep. The second technique combines historical sensor data with real-time sensor data in order to determine whether to preheat the home or start heating after the occupant has arrived. The third technique allows the system to move far away from the temperature set point when the confidence that the home will be

unoccupied is high. This is called *deep setback*.

The paper uses Hidden Markov Models to determine the current presence state of a home: the state can be *Away*, *Active* or *Sleep*. The determination of when to turn on the heating system is based on choosing an optimal preheat time based on the heat rate of the equipment as well as the expected time of arrival of the occupant.

The performance of the proposed solution is evaluated using the EnergyPlus simulator software [9]. The structure and layout of the house is entered as input to the software, which can then run simulations based on different climate zones. The simulation is validated using a real-life deployment of over 100 sensors in a residential type building. The algorithm is compared against an optimal algorithm that achieves perfect savings by only heating when necessary. Two evaluation metrics are used: *energy savings* and *miss time*. Depending on the climate zone, simulated energy savings ranged from about 25% to about 47%. This study applies a smart heating solution to an existing HVAC system. This study provides some of the fundamental on which our work builds, at the same time it differs from our work in several ways. Whereas the study by Lu et al. works with a heating system that can heat with several different speeds, our solution uses radiators, that only have a single speed. This removes some complexity in determining when to start heating. Also, the solution uses an explicit heat rate, whereas our solution has an implicit heat rate contained within the office temperature model. This solution differs from our own in that they use a different model to determine the state that the building needs to be in. They use a set of constraints in the form of equations for which some optimal solution exists, and that convert this into a state that the heating system needs to be in. While we heat offices using radiators, their approach instead heats the concrete of the building by hot water running through pipes in the concrete.

2.3.3 MPC-based Smart Heating Solution - Sturzenegger et al.

Sturzenegger et al. [48] propose a solution based on Model Predictive Control (MPC) [43], a technique that "uses a mathematical model of the building and predictions of disturbances over a given prediction horizon for defining an optimization problem that is solved such as to maintain thermal comfort for the occupants while minimizing some objective (e.g. energy use of monetary cost)." Its goal, like the others, is to keep the occupants comfortable, i.e. reach a certain minimum temperature, while also saving energy. It achieves this by defining a so-called MPC problem, a set of equations with variables representing states, inputs, predictions and outputs. These equations can represent constraints, for example stating that the temperature needs to be between a minimum and maximum value. This constraint would look something like this: $y_{min} \leq y \leq y_{max}$, where y is the temperature, an output variable. Ensuring that all the equations are valid, and finding the right values is then the means by which the system is controlled.

The technique is evaluated on a 6000m^2 office building in Switzerland. The heating and cooling system works mainly with a series of pipes that run through

the concrete of the building, carrying hot or cold water. This technique is also known as a thermally activated building system (TABS) [38]. The results show that the specified temperature comfort range was adhered to for all but one day, an exceptionally warm day in June. The back-up control system, to which the system would revert back to in case the primary control system should fail, was never used. No complaints from the building occupants were reported. The technique was also tested using simulation software, EnergyPlus [9]. The proposed solution used 17% less energy than rule based control techniques. In August the MPC technique resulted in significantly fewer violations of the acceptable temperature range than the rule based control technique.

2.4 Background on Clustering Time Series

We include some background on clustering data, and particularly time series data, in order to facilitate a proper understanding of the design and workings of the feature engineering framework, as described in Section 3.2. There is an excellent survey performed by T. Warren Liao, we include its most important parts here.

2.4.1 Clustering of Time Series Data - Liao

Liao [39] presents a survey on clustering time series techniques. Most existing clustering techniques are applied to static data, i.e. data that does not change over time. Han et al. [31] identify five different categories in clustering techniques for static data: partitioning methods, hierarchical methods, density-based methods, grid-based methods and model-based methods.

Partitioning methods construct k partitions of the data, with each partition representing a cluster. Partitions can be *crisp* if each data point belongs to exactly one cluster, or *fuzzy* if a data point can belong to multiple clusters. Two well-known examples of crisp partitioning methods are k-means [42] and k-medoids [32]. Similar algorithms exist for fuzzy partitions, and are known as fuzzy c-means [26] and fuzzy c-medoids [35]. These algorithms work well for spherically shaped clusters.

Hierarchical clustering methods work by creating a tree of clusters, and they come in two forms: agglomerative and divisive. The agglomerative methods start by placing each data point in its own cluster, and then merging clusters into bigger clusters until either the desired number of clusters is reached, or there is one big cluster. The divisive methods start from one big cluster, and then divide it into smaller clusters. The main drawback of hierarchical clustering is its inability to adjust clusterings after they are made.

Density-based clustering methods work by increasing the size of a cluster so long as the number of data points per cluster exceeds some value. An example algorithm is DBSCAN [28].

Grid-based methods work by dividing the object space into a number of cells, forming a grid. All clustering operations are then performed on this grid. An

example of a grid-based clustering algorithm is STING [50], which uses multiple layers of rectangular cells, each layer with a different resolution.

Model-based methods create a model for each cluster, and then try to fit the data to each the model. This is done in either a statistical fashion, or using neural networks. An example of statistical model-based clustering is AutoClass [24], using Bayesian statistics to estimate the number of clusters. Two examples of neural network-based clustering are competitive learning, such as ART [27] and self-organizing feature maps [34].

Time series can be:

- discrete-valued or real-valued
- uniformly sampled or non-uniformly sampled
- univariate or multivariate
- of equal length or of unequal length

Clustering algorithms designed for working with time series can either work with raw data, i.e. the raw-data based approach, or work with some derivation of the data, i.e. a feature-based or model-based approach. For the raw-data based approach the modification to the algorithm for static data usually lies in the way the similarity between two time series is computed, i.e. the similarity metric. The feature-based and model-based approaches first convert the data into something that can be used by the conventional clustering methods. Figure 2.2 shows these three different scenarios.

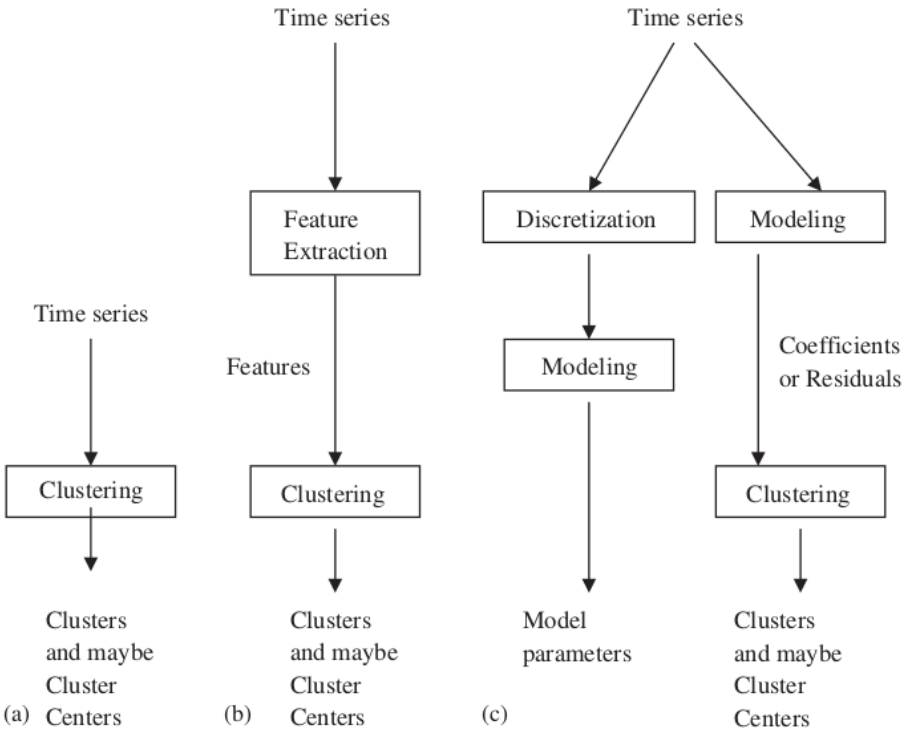


Figure 2.2: Three time series clustering approaches: (a) raw-data-based, (b) feature-based, (c) model-based. Image and caption adapted from [39].

The k-means algorithm has as the objective to minimize the total distance from each data point to its cluster center. It is an iterative method, where the number of clusters is prespecified. The initial cluster centers are randomly picked points from the dataset. Then, for each data point the nearest cluster is determined, and the data point is assigned to that cluster. Once this is done for all data points, each cluster computes its mean, which will be the cluster's new position. This repeats until the algorithm converges. Convergence in this sense meaning that each point is assigned to its nearest cluster. It should be noted that k-means is a heuristic, and is not guaranteed to find a global optimum.

A distance measure that is often used for k-means is the Euclidean distance. In N dimensional space, the Euclidean distance between points a and b is given by the following equation:

$$d(a,b) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}. \tag{2.1}$$

This can be applied to a time series by taking the time series of length M to be a single M dimensional point.

Chapter 3

Design

Two systems are designed: a smart heating system that aims to save energy while keeping the occupants comfortable, and secondly, the feature engineering framework that is used as a tool to improve the quality of the smart heating solution. This framework aims to provide support in finding the best model of the room temperature by finding patterns in the data. These patterns are then expected to be translated into new features by the data modeler, and these new features have the potential to improve the quality of the model.

3.1 Smart Heating System

The smart heating system is comprised of two components, as depicted in Figure 3.1.

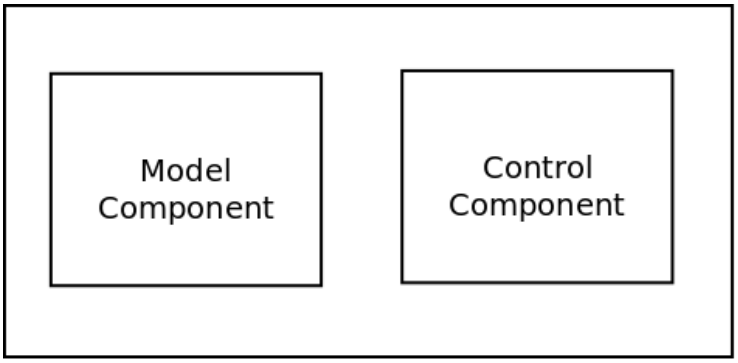


Figure 3.1: Components of the smart heating system.

The **Model Component** is responsible for creating the model of the room temperature. This model allows the room temperature to be predicted, given some knowledge of the environment and historical data. This component is explained in more detail in Section 3.1.1.

The model as created by the Model Component allows the **Control Component** to do its job, which is to determine the state of the actuators. Given some current state and a predicted future state, the Control Component determines the state in which the actuator should be set at the current moment. Section 3.1.2 explains the Control Component in more detail.

3.1.1 Model Component

The goal of the Model Component is to provide some model of the environment that the Control Component can use to determine the right state of the actuators. There are many possibilities in choosing the kind of model, and for this project we choose to go for a linear regression model. There are several reasons that drive this decision. The first reason is a practical one: the requirements of this project as put forward by the collaborating company state that Apache Spark is to be used for data processing. The motivation behind using Spark is that the number of buildings that are expected to be equipped with the smart heating system is quite large, resulting in a large amount of data. This is one of Spark's main use cases: fast, large-scale data processing. Processing time on a server is valuable, and especially in a system that requires relatively quick control of components based on data processing, it is important that even at a large scale data is processed in a timely fashion. Once the linear regression models have been made, using them for prediction is very fast, because only a single equation has to be solved. This is unlike other modeling techniques such as neural networks. The actual creation of the linear regression models takes more time, but this only has to be done sporadically and can be scheduled to optimally use server resources, as it is not a time-critical task. Spark has a machine learning library called MLlib [4], which includes an implementation of linear regression.

A linear regression model predicts what is called the *dependent variable*, y, using one or more *independent variables*, X. The model then takes the form $y = \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$ for n independent variables [36]. When there is only one independent variable, it is called simple linear regression. When there are more, the full term is multiple linear regression. The weights to this equation can be obtained using a method called Stochastic Gradient Descent. This is an iterative optimization method that aims to minimize some objective: in our case the objective is to minimize the error function between the data points and the model.

For our purposes the dependent variable is the room temperature inside an office, this is what we want to predict. The independent variables are not as clear-cut, relying in large part on the domain knowledge of the modeler. Part of this thesis is devoted to helping the modeler find these independent variables, or features as they are also called. Initially however, the modeler will have to rely on his understanding of the environment and of the variable being predicted, and then experimenting to see which environmental variables help explain the dependent variable. Experimenting in this sense means that one creates a model with a certain set of features, and then assesses this model's performance. Doing this with different features, leaving some out and including others, gives insight into what role they play with regards to the dependent variable.

In the case where the room temperature is the variable to be predicted, there are some intuitive features that come to mind. The *temperature of the radiator*, for one, is certainly a feature that we expect to contribute to the room temperature. In the winter this will probably be the main source of heat, as we know from experience that without active heating, the temperature of a building will drop if it is cold outside. This naturally leads us to another feature to consider: *the outside temperature*. There surely is a difference between minus ten degrees centigrade and plus thirty, particularly if the building is not isolated very well, the outside temperature can have a significant effect on the room temperature. The outside temperature is not the only weather aspect that could influence the temperature, the sun is itself an enormous source of heat. One feature that we can extract from this is whether or not the sun is shining, i.e. *cloudiness*. Others could be *sun intensity*, and maybe even the *sun angle* could play a role. The last physical feature we consider is the *presence of occupants*. Not only do human bodies radiate heat, whenever somebody is present they are also likely using electrical equipment such as computers, printers and monitors. All of these electrical devices emit at least some heat.

One feature that we need in order to control the actuators correctly in the control phase, is the position of the actuator. Section 4.1.1 explains in more detail how the actuators work, what is important is that they have an internal valve that we can either open or close, resulting in a temperature difference of the radiator. When we want to control the actuator in the control phase, the model needs to know what the room temperature will be when the internal valve is closed, or open.

Aside from physical features there are a few practical, "virtual features" that are useful. One example is the addition of a *bias feature*, an array where each value is one. This bias term allows for the existence of an intercept other than zero. Without a bias term, the linear model is forced to go through the origin, while this may not necessarily reflect the data well. What this essentially allows for is a "default value" when all the other features in the model sum up to zero. Without a bias feature, the default value would always be zero, including it allows for a default value other than zero. With the inclusion of a bias term the model can cross the y-axis at any point. Other features that we add result from the separation of weekdays and weekends. We expect different behavior for weekends as opposed to weekdays, as people in this particular office building are not present during the weekend. This separation is achieved by having one feature where for weekdays the value is one, and for weekends the value is zero. The feature for weekends has the value one for weekends, and value zero for weekdays. These are also known as "dummy variables" [29].

Not all of these features were available for this project. To summarize, these are the features that are used:

- Outside temperature
- Weather classification (sunny, cloudy, rain, etc.)
- Occupant presence
- Valve position
- Bias

- Weekdays dummy variable
- Weekends dummy variable

The most prominent one that is missing is the radiator temperature. This is due to the fact that the heating system was turned off during a large part of data collection because of the summer. The sun intensity and sun angle were not available for our location.

3.1.1.1 Uniform Data Frequency

When working with time series it is important that the data is regular and uniform, i.e., there are no missing or extra values. These missing or extra values would offset the time series' alignment with other time series, corrupting any analysis performed on it. In reality however sensors do sometimes omit values, or report extra ones. This can be caused by a variety of reasons, from hardware related issues such as a low battery to environmental issues such as radio signal noise. Whatever the cause, it is important that the data is correct. It is for this reason that we perform some preprocessing steps on the data before we use it to create our model.

The way we repair missing data is by interpolating between the closest known values. If the feature is a physical feature with real numbers such as temperature, the interpolation can be conducted in a linear manner. If the feature is more abstract, such as presence or weather classification, the neighbouring values within the time series can be duplicated where necessary. This process is applied to ensure that the time series can be properly handled. Techniques exist to deal with time series of uneven length, but in this interpolating the data will do the job without adding extra complexity to the algorithms.

3.1.1.2 Data Normalization

Aside from making sure that the frequency of the data is uniform, for linear regression it is also desirable to normalize the data. Normalizing the data ensures that a feature that has a naturally much larger range of values does not overpower other features with smaller values. There are several ways to normalize data, for example, one could take the minimum value and the maximum value of a time series and then map those and everything in between to zero and one. The problem with this method is that whenever one would want to use this model to predict a value, the input data to the model, i.e. the feature data, also needs to be normalized. It is possible, although with a large enough dataset unlikely, that the new value falls outside the range of the min and max computed earlier, resulting in a value outside the range zero and one. A method that does not have this problem is using the z-score, where the mean and standard deviation of the time series are computed after which each value of the time series is mapped into the number of standard deviations it deviates from the mean. Should a new value now fall outside the range of the min and max of the original data, this is handled automatically because the number of standard deviations it differs from the mean will simply be higher.

3.1.2 Control Component

The Control Component is responsible for ensuring the optimal state of the actuators, i.e. it is the entity responsible for ensuring a comfortable room temperature for the office, while minimizing the amount of energy spent on heating. This is achieved using the model of the room temperature as created in Section 3.1.1. There is some look-ahead period, for example three hours. This look-ahead period is needed because of the non-immediate nature of heat dissipation, i.e. if we change the valve position it takes time for that to have an effect on the room temperature. The look-ahead period mostly depends on how fast the change of valve state takes full effect on the room temperature, but also on the expected difference between the temperature set point and the lowest temperature.

In order to predict the room temperature at some point in the future, we need predictions of what the features will be at that point in the future. So, we need a prediction of the occupant's presence, of the weather, and of the radiator temperature. The uncertainty of all of these predictions increases as the look-ahead period gets longer, so care should be taken not to make it too long. The last parameter we need is the temperature set point, which represents the lowest acceptable, but comfortable to work in, room temperature.

With all these building blocks in place, we can start describing how the actual control part works. At time T, we want to ensure the set point at time $T + l$, where l is the look-ahead period. Given a forecast of the features far ahead enough to reach the time $T + l$, we enter the temperature set point as y in the model's equation, and enter all features except for the valve position. Now there is one equation with one unknown variable, which can be determined, so then we have the required valve position for time $T + l$. However, we also need the valve positions for each epoch between time T and time $T+l$ in order to actually reach the temperature set point at time T. Then we obtain the valve positions for each epoch in this time frame by interpolating the desired temperature at time $T + l$ with the current temperature. Together with the predictions of the other features we can then, for each epoch between time T and time $T + l$, determine the valve position, resulting in a series of required valve positions that ensure the right temperature at time $T + l$.

If there would always be presence, this would be it for the control part. We would simply maintain the temperature set point at all times. However, whenever there is no presence, there is no need to maintain the temperature set point, providing the opportunity to save energy by changing the valve position to save mode, reducing the room temperature. This can be achieved by having the default state of the valves be the "save position", and only changing it to the non-save position whenever it is necessary. The risk associated with this approach is that the temperature can drop too low, to such a degree that the office cannot be heated up within the timespan of the look-ahead period. A second, lower-bound set point is used to deal with this. This lower-bound is simply a temperature that we should not get under.

3.2 Feature Engineering Framework

The goal of the feature engineering framework is to allow for the creation of the best possible model of the room temperature. Ultimately we want to use it to discover new features that can be used by the model to increase the accuracy of the temperature prediction. This is perhaps best demonstrated with an example. Say there is a model that predicts the room temperature based on the occupant's presence, and the outside temperature. Then the goal of this framework is to find additional features, such that the prediction is improved. For example, using the techniques described in this section one might discover that most offices on the south side of a building are always warmer than the offices on the north side of the building. This might be a good indicator that the influence of the sun should be taken into account with regards to the model, so the sun intensity could be considered as a new feature for the model to use.

The feature engineering method consists of two major steps:

1. Visualization of the data
2. Clustering of the data

The visualization part is important, because we are trying to gain as much insight into the situation under investigation as possible. Seeing as the feature engineering process will be carried out by humans, it makes sense to make full use of one of our best sensing systems: visual perception. The human visual perception system has evolved over millions of years to keep us safe from predators and spot berries and nuts in the bushes. We can process a lot of information with our eyes, and that is exactly what we need in order to present the large amount of data that we have.

The second step, clustering the data, aims to really expose relations between offices. What we essentially want to do is find common patterns of behavior between offices. In other words, we are looking for offices that are in some way similar to each other. The way we will look for similar offices is by means of *clustering*: given some data set and a measure of similarity, provide a set of partitions such that the distance between items within a cluster is minimized. If this clustering is combined with the visualization of the data, this makes for a powerful combination of tools. We consider three different methods to cluster the offices.

3.2.1 Clustering

There are a number of different ways in which the offices can be clustered. One of the most straightforward methods is to cluster the temperature time series of the offices. Finding similar trends in temperature development and grouping them together in a cluster can give insight into the similarity of offices.

The second method is slightly more complicated. For this method we assume that all offices are the same, and can be explained by a single model. This assumption is unlikely to be true, but making it allows us to demonstrate that the offices are in fact different, because the resulting model will not explain

the behavior of the offices well. This single model is then used to predict the temperatures of the offices, resulting in time series of predicted values. If we then compare these predicted values against the actual values, the ground truth, we obtain what we call the prediction error. This is a signed number indicating the degree to which each office differs from the generic model. Clustering on this prediction error groups together the offices that show an equal deviation from the generic model.

The third method of clustering works by creating a model of the room temperature for each office individually, and then clustering on these models. So the offices with similar models will be grouped together in the same cluster.

The first two clustering methods cluster time series data, whereas the third clustering method clusters based on multidimensional points. We can actually use the same clustering algorithm and similarity metric for both use cases.

3.2.1.1 Clustering Algorithm

When deciding what clustering algorithm to use, there are several aspects to consider. One of them is the type of algorithm, whether it be a partitioning method like k-means, a hierarchical method or some other method. Closely related to the decision of the algorithm is the similarity metric. The similarity metric quantifies how similar two data points are. A data point in this sense can be a time series, or a regular multidimensional point. The choice of similarity metric can depend on what kind of data is being clustered, the best metric for time series is not necessarily the same as the one for static data. Another decision that has got to be made is whether to work with the raw data, or with some derivation of the data.

The advantage of working with the raw data is that no details are lost, everything is taken into account for the clustering. The inherent drawback of this is that there is more data to process, making it more computationally expensive. The opposite is true for the derivation-based methods. They usually consist of much less data, while compromising details of the original data. Because at this point maintaining the highest possible level of detail of the original data is more important than the computing time, we decided to work with the raw data instead of working with some derivation of the data.

When clustering time series data, there is one similarity metric that is generally considered the best fit for most use cases: Dynamic Time Warping [25]. It is very good at dealing with phase-shifts between two time series. However, its major drawback is that it is computationally expensive. This can become problematic, particularly when working with large amounts of data. Also, while it is good at dealing with phase-shifting between time series, it is not always desirable to correct for this. Take as an example two groups of offices that have a phase shift of one hour, so one group reaches its peak temperature one hour before the other group. It is possible that DTW would put these two groups in the same cluster, accounting for the phase shift. However, it might actually be desirable to separate the two groups, as the phase-shift could indicate an underlying feature that we want to expose. For example, it might be that the angle of the sun is causing the one hour delay of reaching peak temperature,

in which case it might be desirable to have some information about the office locations and orientation as a feature for the model. It is for these two reasons that we decided not to go with what may be the obvious choice of similarity metric, but rather look for a metric that better fits our needs.

The next choice for similarity metric that would make sense is the Euclidean distance. It is not very computationally expensive, and the way it works is quite intuitive. The question that remains is: is it appropriate for time series data? As it turns out, it is appropriate. Computing the Euclidean distance between two time series is no different from computing it for an ordinary multidimensional point. In fact, a time series can be seen as a high-dimensional point, where each epoch of the time series represents a dimension.

Now that the most appropriate similarity metric has been established, it is time to determine the most appropriate clustering algorithm. There are a lot of options when choosing a clustering algorithm. One of the most popular methods is k-means [42], a partitioning algorithm. One of the big advantages of the k-means algorithm is that it has a linear time complexity, $O(n)$, whereas most hierarchical clustering algorithms have a quadratic time complexity, $O(n^2)$. The main disadvantage of k-means is the fact that the number of clusters k has to be predetermined. Another alternative to k-means is the k-medoids algorithm. This algorithm is more robust to outliers and noise in the data. However, like the hierarchical clustering algorithms it has a time complexity of $O(n^2)$. It is for this reason that we choose to go for the k-means algorithm for our clustering purposes.

Determining k The Elbow method [49] is used in order to determine the number of clusters that should be used for the analysis. This method works by computing the average variance within each cluster for different amounts of clusters. For example, taking the number of clusters k in the range $[1, 10]$, the average cluster variance is computed for each k. Plotting this results in a line where the variance should be highest for $k = 1$, and theoretically would be 0 for the case where $k = n$, n being the number of data points. Figure 3.2 shows an example of such a plot. This reduction in variance as the number of clusters increases can be explained by the fact that the k-means algorithm groups together similar data points, thus reducing the variance as more clusters come into existence.

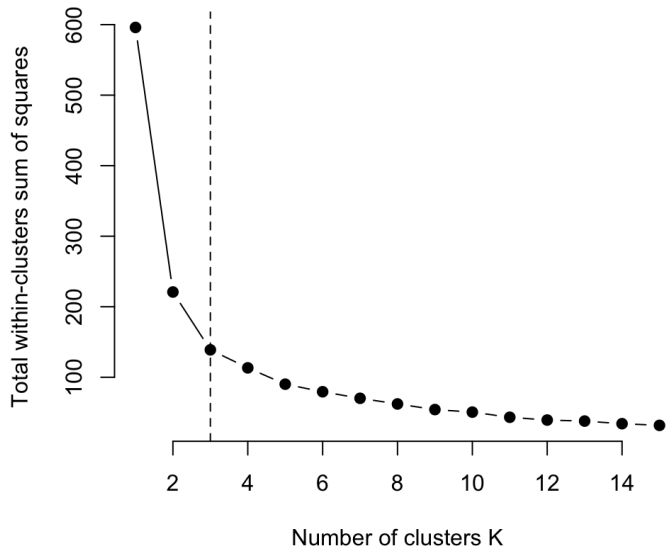


Figure 3.2: An example of applying the Elbow method. Figure adapted from [5].

The way the Elbow method works is well defined for static data, but we could not find any works that applied it to time series data. The main problem in case of time series data is computing the variance between a collection of time series. The variance within a single time series is well-defined, as this is the same computation as for the variance of any static data. However, the computation of the variance *between* two or more different time series is, to the best of our knowledge, not so well-defined. Computing the variance for static data is straightforward, as demonstrated in equation 3.1. In this formula x is the collection of data points, μ is the mean of x, n is the number of data points and σ^2 is the variance. If this were to be applied to time series data, x_i would be a time series. The resulting variance would then also be a time series, which is not suitable for our purpose. What we need is a single value indicating the variance between two or more different time series.

$$\sigma^2 = \frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n} \tag{3.1}$$

We created a slight adjustment to the conventional variance that works with time series. For each epoch of the time series the variance is computed over all the time series. So if there are three time series, T, U and V all of size N, then the variance is computed for the values T_1, U_1 and V_1, then for the next epoch: T_2, U_2 and V_2, eventually resulting in N variances. The average is then computed by summing all these variances, after which they are divided by N.

This allows us to get some insight into the variance between time series, thus allowing us to determine the optimal number of clusters.

Chapter 4

Deployment and Implementation

In this chapter the specifics of the deployment and implementation of the smart heat system are presented. The general context of the solution is explained, as well as the hardware that has been used, the deployment topology and the implementation details.

The smart heating solution is deployed in one of the buildings of the University of Groningen, the Nieuwenhuis building [1], as depicted in Figure 4.1. Located in the city of Groningen, it contains about one hundred offices where researchers of the pedagogy department work. Figure 4.2 shows the geographical location of the building.

Figure 4.1: Aerial photo of the Nieuwenhuis building [11]

This building was chosen for several reasons. It is a relatively old building, so it has not been equipped with an advanced Building Management System (BMS) [33]. The building is heated using radiators, which has several implications. Since the temperature of the radiator is set by the office occupant, there is likely room for improvement with regard to saving energy whenever the occupant is not there. It is unlikely that occupants are aware of the optimal moments to turn the heating down when they anticipate their departure from the office, and it is at best unpractical for occupants to turn up the heating in anticipation of their arrival. Even if they were able, it would be entirely understandable if they were unwilling to shift their focus from their work to saving energy, multiple times a day. While a smart heating solution may also

be unable to determine the exact optimal moments to turn the heating up and down, it can make an estimation based on the office context, and is not hindered by any of the other issues. Another upside to using radiators is the ability to individually control the temperature of offices. While other heating systems may heat an entire building to a uniform temperature, the ability to control individual offices has the advantage that the specifics of each office such as occupant behavior, office orientation, can be taken into account in creating a control scheme for that office. In other words, we are able to adjust the temperature of an office based on the properties of that office. This allows us to provide a much more tailor-made solution, with the potential to save more energy.

In this chapter we present the details of how a smart heating system has been deployed in a real environment, the Nieuwenhuis building. In the next sections the design decisions with regards to infrastructure are discussed.

Figure 4.2: Map of the Nieuwenhuis building in Groningen, obtained from Google Maps.

4.1 Hardware

Three different devices are used for the smart heating project. This is excluding hardware used for the back-end such as servers. Sensors and actuators are used to obtain environmental context and control the room temperatures. Gateways are used to collect the sensor data and send control commands to the actuators.

4.1.1 Sensors

In order to realize the smart heating solution, one type of sensor and one sensor/actuator combination is used, both are manufactured by a company called Kieback & Peter [14]. The first is the room sensor, which measures the room temperature and the presence. The sensor/actuator, which is mounted on the radiator, measures the temperature of the radiator. It also has the ability to adjust the heat that is produced by the radiator, both by manual adjustment of the knob position, and by changing the internal position of the valve to either

"save" mode or "normal" mode. When in "save" mode, the room temperature should theoretically be 4°C lower than when the valve is in "normal" mode. The exact difference also depends on environmental factors such as whether or not the sun is shining into the office.

Figure 4.3: Actuator (MD10-FTL-HE)

Both sensors communicate using the EnOcean protocol [8], a wireless, light-weight communication protocol. It is often used in combination with energy-harvesting hardware, such as the two sensors used for this project. The room sensor harvests energy using a solar panel, obtaining enough energy from artificial lighting. The actuator harvests energy from the heat of the radiator, using a thermoelectric generator inside the actuator. Both sensors contain a small battery to bridge periods of time where the source of energy is not available. However, the batteries are meant as a backup, and eventually the sensor will stop functioning if the energy source is not available, i.e. prolonged darkness for the room sensor or prolonged absence of heat from the radiator. The sensor will resume functioning as normal when the energy returns.

Figure 4.4: Room Sensor (RPW401-FTL)

4.1.2 Gateways

The gateways function as a regional hub for the room sensors and actuators. They receive the packets from the sensors to which they are paired, and send packets to the actuators in order to control them. Fulfilling this role are Raspberry Pis [18], as depicted in Figure 4.5. These devices offer a reasonable amount of computing power while consuming little power. They are equipped with an Ethernet port in order to connect to the Internet, and have USB ports that allow us to attach an Enocean USB gateway [10].

The gateways are powered using Power over Ethernet (PoE). This was a more cost effective option than powering the gateways using traditional power sockets,

as these were not available for use in the locations where the gateways were to be deployed. Power is injected in the central server room, from where the power and Internet signals are transmitted to each gateway location. Here the cable is split into a power and Ethernet cable, using a PoE splitter. This allows the right cables to be connected to the gateway.

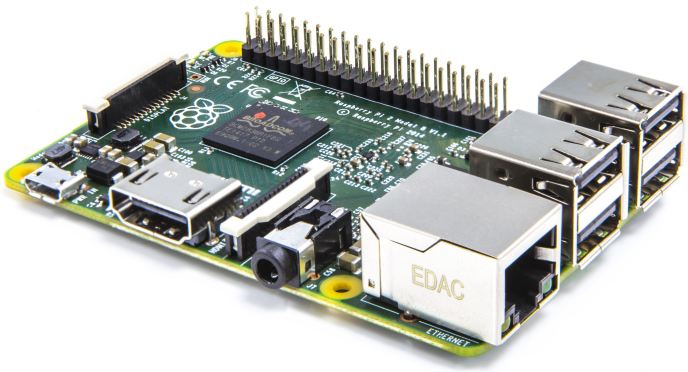


Figure 4.5: Raspberry Pi

4.2 Deployment Topology

One of the variable factors in deciding on the deployment topology was the positioning of the gateways. Seeing as the sensors communicate with a wireless protocol, the question was how far the signals would reach in an office environment. Having fewer gateways is desirable as it saves on purchasing and installation costs, as well as any future maintenance costs. However having too few gateways will result in messages from the sensor being lost, so there is a trade-off between costs and reliability in the coverage of the network. We decided that the best way to decide on the locations of the gateways was to do some in-field testing of the signal strength of the hardware.

Figure 4.6 shows the entrance of the building on the ground floor. For this location we tested two possible gateway configurations: one where there is a single gateway at point A, and one where there are gateways at points B and C. Using the DolphinView Advanced software [7] we analyzed the signal strength of the transmitted packets and looked out for any packet loss. When placing the single gateway on location A, there were lost packets for one of the offices we wanted to reach for this location. When placing the two gateways on this location, at points B and C, there were no lost packets for any of the offices in this area. We moved through the building testing different configurations of the gateways in order to figure out what worked and what did not, resulting in the eventual deployment topology.

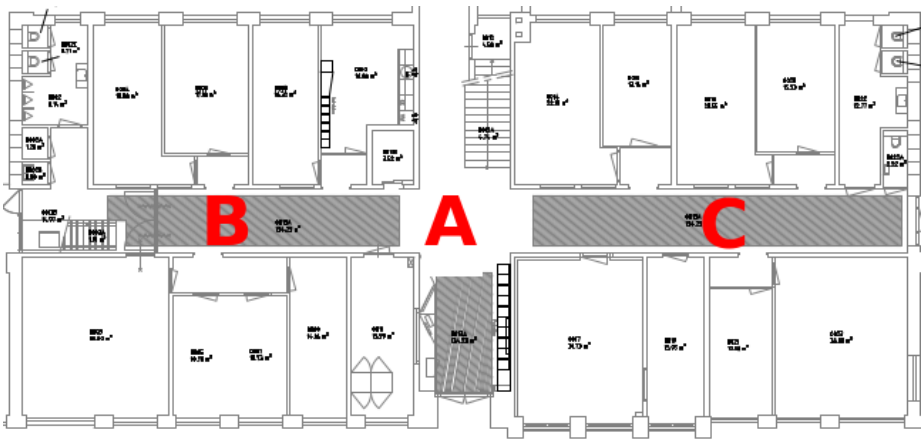


Figure 4.6: Entrance of the building, indication of tested gateway positions.

4.2.1 Testing for signal interference

Because the Nieuwenhuis building is part of the University of Groningen, planning and installation of the hardware was done in collaboration with the university's technical department. One of the requirements from their side was that the wireless signals of the sensors must not interfere with any existing systems that reside within the building. To ensure this was not the case we performed a frequency analysis to see if the frequency band that our sensors use, the 868,3 MHz band, was used or not. The results showed that the frequency band was almost completely unused, except for one or two locations where it was used very lightly. Seeing as the EnOcean protocol is very lightweight, and the frequency band was practically unused, this requirement was met and the installation could proceed.

4.2.2 Back end

Once a sensor value reaches its gateway, the value is then pushed to the back end. This project is conducted in collaboration with the Sustainable Buildings company, and as such we use their back end for data storage. Figure 4.7 displays a schematic of the back end. As mentioned before the sensors send their data to their assigned gateway. The gateway performs some processing on the data after which it is pushed to the message queue, in this case a RabbitMQ [17] server. The items in the queue are then consumed by the data collector, which stores the items in a Cassandra [20] cluster.

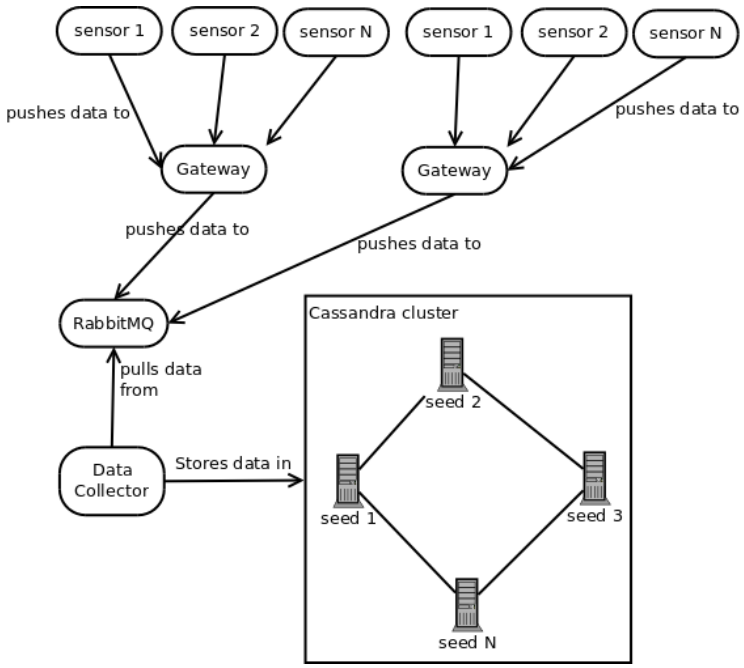


Figure 4.7: Back end used for data reliable and scalable storage.

This design has several advantages. By having the message queue act as a buffer, the data source is decoupled from the data storage. This allows data sources to come and go as they please, while also ensuring that the gateways do not have to wait on a response from RabbitMQ, seeing as it is an asymmetric operation. Cassandra is used as the database. One of Cassandra's main use cases is handling time series data, which is exactly what is required for this project. Cassandra has a peer-to-peer design, connecting nodes in a ring formation. This, among other things, allows for linear scalability and no single points-of-failure.

4.3 Implementation

In this section we look at the specifics of how the design is implemented. To get a better understanding of the environment in which the project is carried out, we go over the technologies that have been used to realize the project. Then, the specifics of how the software was constructed for both the data collection part and the data analysis part will be presented. Providing these implementation details may prove useful to people working with these technologies in future projects.

4.3.1 Technology Stack

Figure 4.8 shows the technologies that are used for this project. These are all technologies that the SustainableBuildings company currently works with, which

is why they are used for this project. We very briefly describe each technology.

Figure 4.8: The technology stack used for this project.

Scala [21] Scala is an acronym for "Scalable Language". Used by many companies, among which are Twitter, LinkedIn and Intel, it is an integration of functional and object-oriented language concepts. It runs on the Java Virtual Machine (JVM), and because Java and Scala classes can be mixed freely, all libraries and frameworks that are available for Java can be used with Scala. Its encouragement of using immutable state makes it easier and safer to write performant multi-threaded code.

Git [12] and GitHub [13] Git is used as the version control software. It is a distributed version control system with a small footprint and fast performance. It is used by many of the largest software companies, including Google, Facebook, Microsoft, Twitter and Netflix. GitHub is used to efficiently coordinate collaboration between team members by keeping track of issues and commits.

Cassandra [20] Cassandra is used as the database technology. Being a NoSQL database, it provides scalability and high availability without compromising on performance. It achieves fault tolerance by replicating data over multiple nodes.

RabbitMQ [17] RabbitMQ is a message queue, implementing the Advanced Message Queuing Protocol (AMQP). It acts like a buffer between data producers and data consumers. Compared to other message queue technologies it is easy to setup, and provides broad possibilities for setting up different routing topologies.

Docker [6] Docker is a virtualization technique that packages an application and its dependencies in a so-called container, and shares the host operating system between containers, resulting a light-weight architecture of virtual instances. Docker containers have the advantage on running anywhere where Docker is supported.

etcd [22] etcd is a distributed key value store used for shared configuration and service discovery.

Weave [23] Weave creates a virtual network that connects Docker containers across hosts, and enables automatic discovery.

Spark [3] Apache Spark is a processing engine built for large scale data processing. Used by large companies such as Netflix, Yahoo and eBay, it is one of the most popular big data analytics tools.

Jenkins [2] Jenkins is an automation server that provides support for building, deploying and automating any project. It can be used for continuous integration and continuous delivery, and has hundreds of plugins available.

4.3.2 Driver / Collection Software

Driver software had to be implemented in order to communicate with the sensors. The input to the software is a stream of bytes, representing the packets as received by the EnOcean USB gateway. There are a number of different types of packets, ranging from packets with a one-byte payload (1BS packets), to packets with a four-byte byte payload (4BS packets), to packets used for the teach-in process used by sensors: Universal Teach-in EEP-based (UTE) packets. EEP stands for Enocean Equipment Profile, this is a description of the data that a particular sensor can send.

Sensor data is identified using two UUIDS [19]. A UUID, universally unique identifier, is a 128-bit value generated using random numbers. One is called the *instance-id*, and is used to represent the physical sensor itself. The other is called the *sensor-id*, this is used to identify the type of phenomenon being sensed, such as light or temperature. The combination of instance-id and sensor-id uniquely identifies a time series of sensor data.

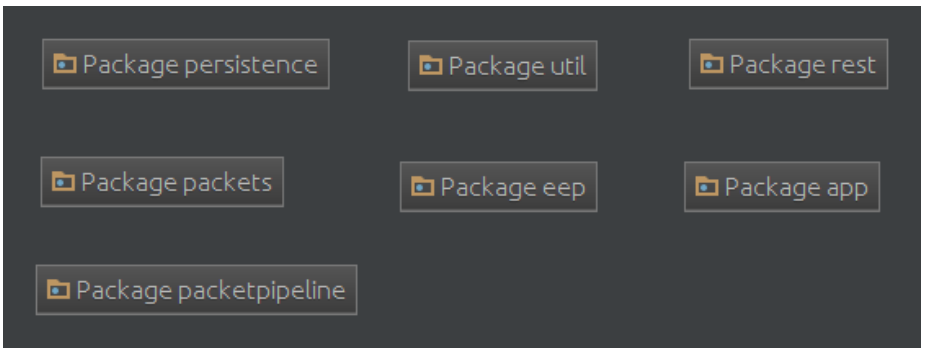


Figure 4.9: Package diagram of the driver software.

Figure 4.9 displays the packages that are part of the driver software. We briefly describe what each package contains in terms of functionality and classes.

Persistence package The **persistence package** contains several classes that deal with data that needs to be stored (semi-) persistently. For example, the physical sensors have a unique identifier as provided by the manufacturer. This needs to be mapped to the correct instance-id. Similarly we need to store the EEP of the sensor, i.e. the description of how to parse the data, this is the responsibility of the *EepStoreActor*. The actuators require a response to each packet that they send, this response contains the state that the actuator should be in. It is the *ActuationPacketStore's* job to coordinate the correct response to the actuators. Finally, in order to push the sensor data to RabbitMQ we need to store objects to which the data can be sent. This is the responsibility of the *PushSourceStoreActor*.

Packets package The **packets package** contains the different types of packets that are used. The *Packet* class is a super class to all packets. The *UteTeachInPacket* class is used for the teach-in process of the room sensors. The *VldPacket* class is used for subsequent data transfer of the room sensors. The *FourbsPacket* class is used for both teach-in and data transfer of the actuators.

Packet Pipeline package The **packetpipeline package** contains the logic that transforms the stream of bytes, as delivered by the hardware, to packets. It starts at the *ByteStreamParserActor*, which separates the byte stream into usable chunks, and encapsulates it in a Packet class. This packet is then passed on to the *PacketDispatcherActor* which dispatches the packet to the correct actor. These actors still represent a broad range of packets, for example the PacketDispatcherActor can pass packets on to the *FourbsActor*, *FourbsTeachInActor*, *UteTeachInActor* or *VldActor*. Each of these represent a category of sensors or processes. The *PacketOutActor* is used to send packets to sensors and actuators.

EEP package Once the packets have reached the correct actor identifying their type, they can be processed according to their EEP. We use two different types of sensors, and as such have two different EEPs that are supported. The *A52001Actor* is used for the actuators, the *D21002Actor* is used for the room sensors.

Util package The **util package** contains some classes to making working with byte data more convenient.

REST package The **rest package** contains the *ValveControlRest* class, which provides a REST interface that is used to control the actuators. A JSON message can be passed to the REST address, specifying the address of the sensor and the position that it should be set to, as such:

```
{
  "sensorAddress": "0189CD63",
  "savePosition": true
}
```

App package The **app package** contains the code to start the application.

4.3.3 Analysis Software

For the analysis part we use Apache Spark on top of a Cassandra database. The Spark MLlib library is used to create the models and perform the clustering. While Spark is capable of stream processing, we use a batch processing approach as it is appropriate for our use case. Creating these models and clustering the data is not a continuous process. Rather, it is something that is usually done only once per building in order to create the best possible model.

The prediction errors are also stored in Cassandra using a <sensor-id, instance-id> pair. This allows us to create the prediction errors once and then perform multiple analyses on this data, resulting in less processing time. Because the sensor data is stored on a production server and we do not want to unnecessarily burden it with our experiments and analyses, some classes have been implemented that copy the data from the production Cassandra cluster into a local Cassandra instance.

The feature data of buildings will likely differ from building to building. To accommodate easy analysis of different buildings we decoupled the data required for the clustering and model creation from the actual feature data. Whenever one would want to perform the analysis for another building, all that is required is that the feature data be delivered in the form of a matrix. The code that models and clusters the data can then use this matrix independent from how many features are used.

Chapter 5

Experiments and Evaluation

In this chapter four different topics are going to be evaluated. We start off with an evaluation of the data: what the data looks like and how the sensors behave. Then we proceed with evaluating the three different clusterings of the offices. We look at how many clusters there are, and what the clusterings look like. Next we evaluate the room temperature models and their quality. Then there is an evaluation of the feature engineering process, after which the chapter is concluded with a discussion of the results. Ideally we would have evaluated the performance of the smart heating system as well. However, due to delays in the deployment of the project, the warm weather of the spring and summer meant that the heating system was turned off. This resulted in us being unable to test the control solution.

5.1 Data Collection

Before we get into the more advanced analytics, it is worth taking a moment to get some basic insight into the data that was collected from the sensors. Displaying this data is not as straightforward as it may seem, due to the large number of offices. A simple line chart displaying about 85 lines would not convey much information, it would simply be a chaotic collection of lines, as illustrated by Figure 5.1. Instead we plot the data using a custom type of bar chart.

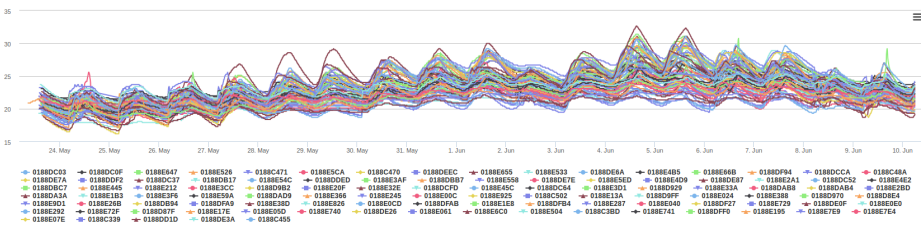


Figure 5.1: Temperature data from the room sensors.

5.1.1 Actuator Temperatures

The actuators are equipped with a temperature sensor, effectively measuring the temperature of the actuator, and giving some insight into the operating state of the radiator. Figure 5.2 displays a chart with temperatures from a subset of the actuators, giving some insight in the behavior of the radiator, and thus the heating system and actuator as well. The figure warrants some explanation. The left side of the figure contains a number of red-lettered sensor identifiers. Each such identifier indicates a row, this row represents the temperature data for that actuator. Whenever an entire row is black, this indicates that there was no data available for that actuator. This can have several causes: either the actuator's battery was empty, the actuator was not paired to a gateway, or the gateway was down. There was one gateway in the building that during the summer was not functioning, which explains at least some of the black rows. Another major cause of black rows is the fact that the data was collected during the summer period. In this period the heating system is largely turned off, not giving the actuators the chance to charge their batteries. This is expected behavior.

The bottom of the figure shows the meaning of the colors in the legend. Temperatures range from about 16 degrees centigrade to almost 36 degrees, although in this particular subset of actuators the temperature does not actually get that high. The colors are determined as follows. The time interval from the start date to the end date is divided in N different pieces, in this case $N = 300$. Then for each piece a vertical line within the actuator's row is drawn. If there is no data available in the particular interval, then the color of the line is set to black. If there is data available, then the average temperature over that time interval is determined, and the color is taken from the legend.

There are a few clear cases where the battery of the actuator runs out, whenever there is data available for a period of time in the beginning, but then at some point the rest of the row is black, that is a strong clue that the battery of the actuator ran out. The more curious cases are those where there is similar behavior, but at some point the actuator will start reporting values for a little while. Perhaps the actuator was charged a little bit, just enough to start reporting values for a while, but then ran out again. The figure shows data from a period of about 2.5 weeks, and the daily patterns are recognizable in the figure. In the beginning the heating system still appears to be turned on, as temperatures reach about 30 degrees during the day. At night the temperature drops to about the room temperature. As we continue in time, we see that the radiators seem to follow the trend of the room temperatures, which makes sense seeing as it is at that point their primary source of heat.

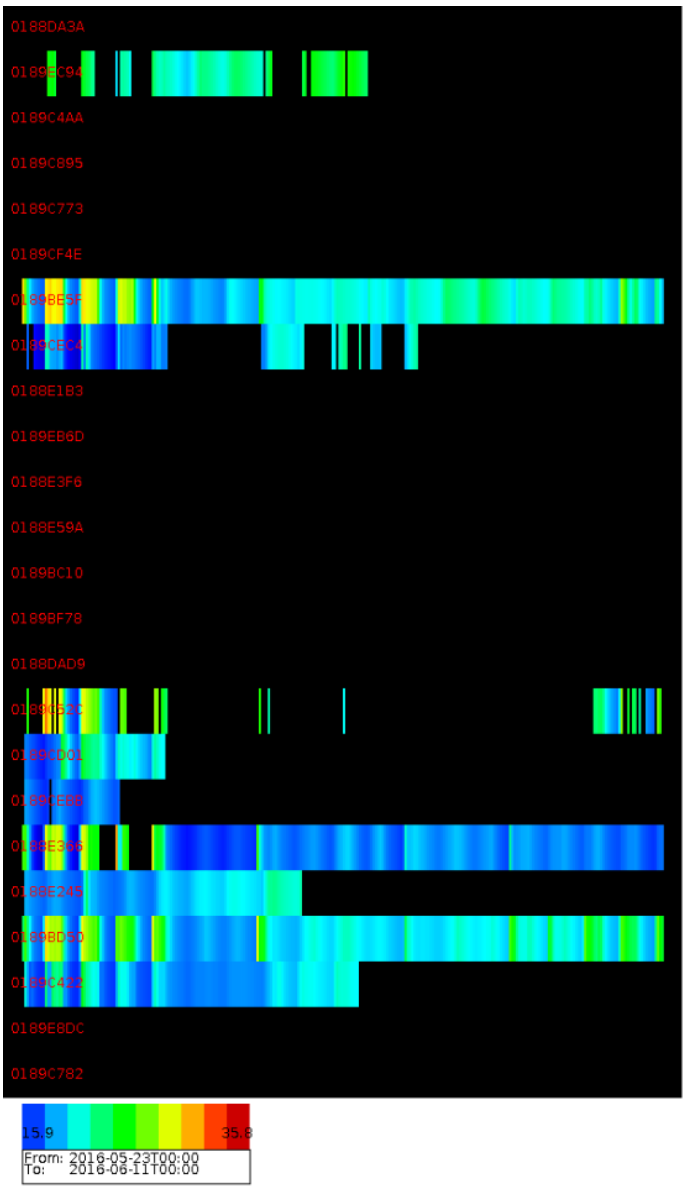


Figure 5.2: Bar chart of actuator temperatures. This is a subset of all actuators. Black indicates that there was no data available for that interval. The red number on each row are the sensor hardware identifiers.

5.1.2 Room Sensor Temperatures

Figure 5.3 shows the temperatures of the room sensors in a similar fashion as Figure 5.2. The time range starts on May 20th and ends on June 10th. The lowest recorded temperature in this period was 16.2 degrees centigrade, the highest temperature was 32.6 degrees. The room sensors are equipped with a photovoltaic cell, and as such transmit data much more reliably than the valves.

The black bars that are still present are due to the one gateway that was not working. We can observe quite some difference between the temperatures in the offices, some are notably hotter than others. Some sensors do show intermittent missing data, at this point it is not yet clear why this is the case. It could be that the office was too dark during these periods for the sensor to charge, thus causing the battery to run out resulting in missing data.

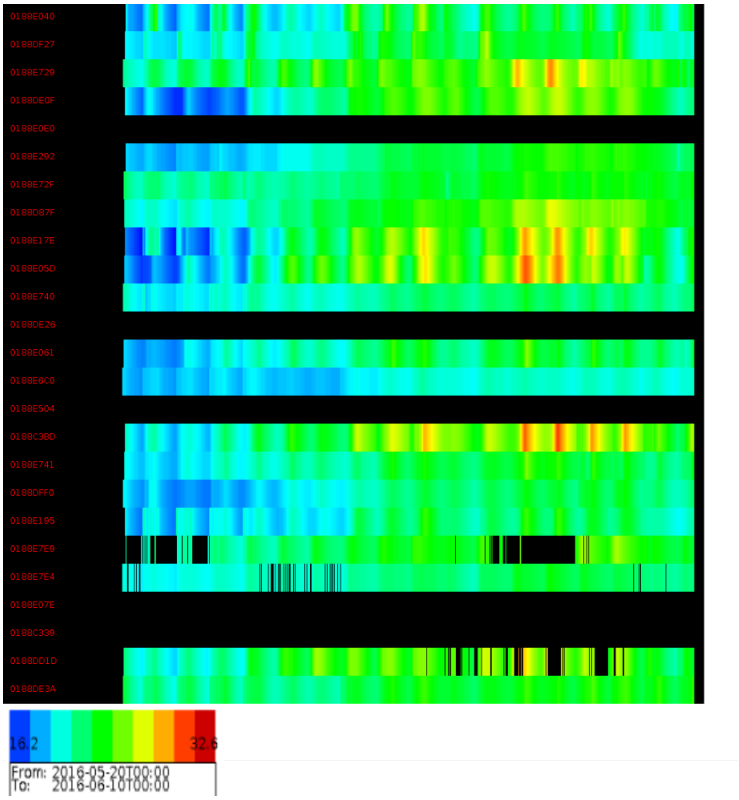


Figure 5.3: Bar chart of room sensor temperatures. This is a subset of all room sensors. Black indicates that there was no data available for that interval. The red strings on each row are the sensor hardware identifiers.

5.2 Office Clustering

We evaluate the clustering using three different clustering setups, as explained in Section 3.2. First, we cluster the temperature time series of the offices. Then, we create one model for all offices, and cluster on the prediction errors of each office. For the third method we create a model for each office, and then cluster on the model parameters. Before we can discuss the results of this, however, we need to determine the number of clusters to use. Then, after discussing the results of each different clustering setup, we can see how the different clusterings compare to each other.

5.2.1 Determining k

Because different techniques are used to cluster the data, there will also be independent assessments of the optimal number of clusters. First, we look at the clustering of the room temperature. Then, we see if the clustering of the prediction errors results in the same optimal number of clusters.

Figure 5.4 displays the graph for the room temperature clustering. The drop is clearly largest when going from 1 cluster to two clusters. The next largest drops occur until 4 clusters, after which each additional cluster reduces the variance only slightly. Placing the elbow is somewhat of a subjective process. It would be justified to place the elbow at either $k = 2$, or at $k = 4$ as the angle between the lines is greatest for these points.

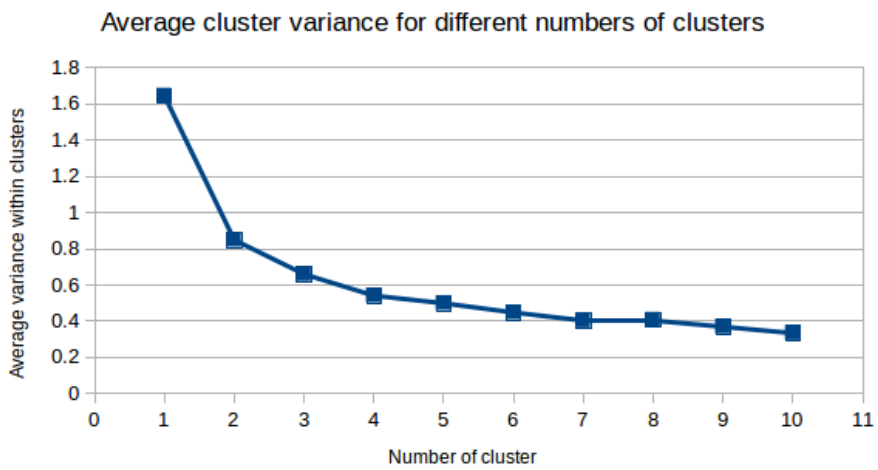


Figure 5.4: Average cluster variance for different numbers of clusters, clustering of room temperatures.

Figure 5.5 displays a similar graph, but now for the clustering of prediction errors. The overall variance is lower than for Figure 5.4, which makes sense as the room temperature values are higher values than the prediction errors. The steepest reduction in variance can again be found when going from one cluster to two clusters. The variance is only reduced a little bit after adding the third cluster, so this clustering would indicate $k = 3$ to be the optimal amount of clusters.

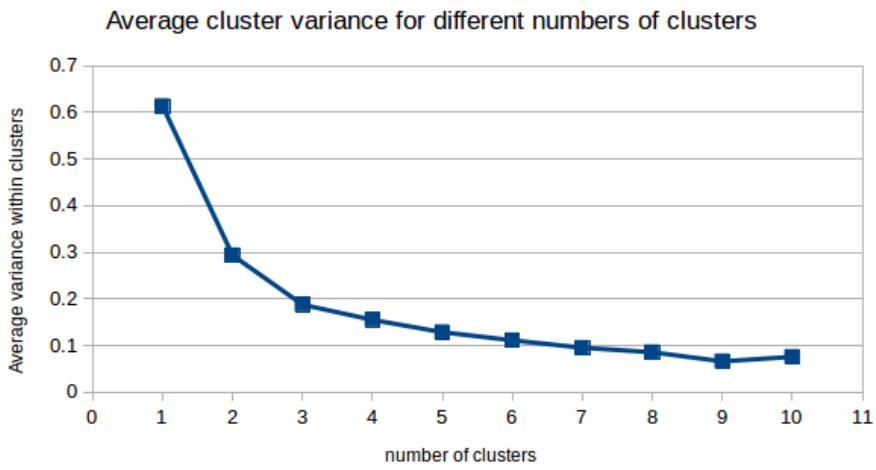


Figure 5.5: Average cluster variance for different numbers of clusters, clustering of prediction errors.

The two independent methods do not necessarily produce the same optimal number of clusters. However, since the second method indicates three as optimal, and the first method either 2 or 4, it is fair to say that since 3 lies in the middle of 2 and 4, that 3 is actually the optimal number of clusters.

This is backed up by the performing the elbow method for the third method: the clustering of the model parameters. Figure 5.6 shows that the third method also produces 3 as the optimal number of clusters.

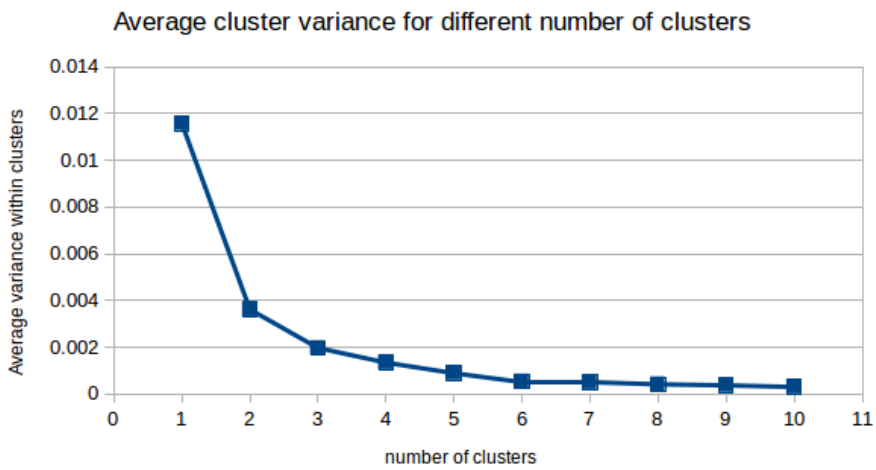


Figure 5.6: Average cluster variance for different numbers of clusters, clustering of model parameters.

5.2.2 Clustering Room Temperatures

Figure 5.7 shows a part of the office clustering.This figure should be interpreted as follows. A circle with a variety of colors is contained in each office for which there is data, and for which the clustering has been performed. The outer colored ring of each circle represents the cluster to which that office was assigned. In Figure 5.7 the clusters are: black, red and green. The inner part of each circle contains a variety of colors. This part is used to gain some insight into temperature of that office. This inner part was constructed as follows. The time series of each office is split up into 360 intervals. The average temperature is computed for each interval, after which a line is drawn on the circle for each interval. Since there are 360 degrees to a circle, the average temperature for each of the 360 intervals can be plotted on the lines. The color is then determined based on the legend that is shown on the top left of the figure. Lines are drawn from the center of the circle to the outside, starting at the bottom of the circle and proceeding counterclockwise. Whenever the color of a line is black, this means that there was no data available for the interval.

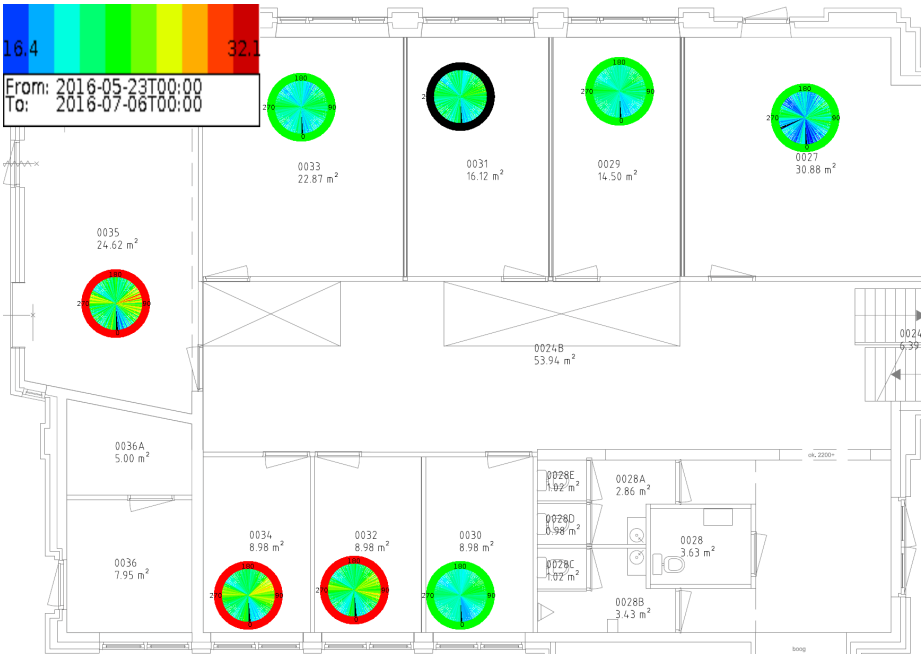


Figure 5.7: Clustering of some offices, using the room temperature. Explanation of the figure is given in section 5.2.2.

This visualization of the room temperature gives some insight into how the temperature of an office behaves, and how it compares to other offices. For example, we can see that the offices that are assigned to the red cluster, on the bottom left are hotter than the offices of the black and green clusters. The leftmost office even reaches a temperature of 32.4 degrees. This ability to visually inspect the temperature of an office helps in determining the quality of the clustering.

5.2.3 Clustering Prediction Errors

Figure 5.8 displays the same office region as Figure 5.7, but instead of clustering on the room temperature the offices are clustered on the prediction error. A detailed explanation of how this prediction error is obtained is given in Section 3.2.1. For this model we used an 80-20 training-testing split. A noticeable difference between these two figures is the larger number of black lines in the second figure. This normally means that the data is missing, as is the case here. There is a good reason for the data not being there, however, because this interval was used to train the model that was needed to create the prediction errors. In order to increase comprehensibility it makes sense to plot the same time period in both figures.

The clustering of this particular area of the building is identical for both clustering techniques. It should be noted that the cluster colors are not the same for the two figures. So while the clustering is the same, the color of the clusters differs. This happens because the clusters are assigned a randomly picked color. Also note that the circles contain some artifacts from the drawing process, this is particularly visible for the parts where many black lines are drawn.

We used these two figures to demonstrate that these two clusterings produce identical results for some areas. This is not the case for all areas. In order to get a better understanding of the similarity of the clusterings, we apply the Jaccard index in Section 5.2.5. Also, while the clustering for this area produced the same result, this also depends on the initialization of the initial clusters in the clustering algorithm. Different runs can produce different clusterings.

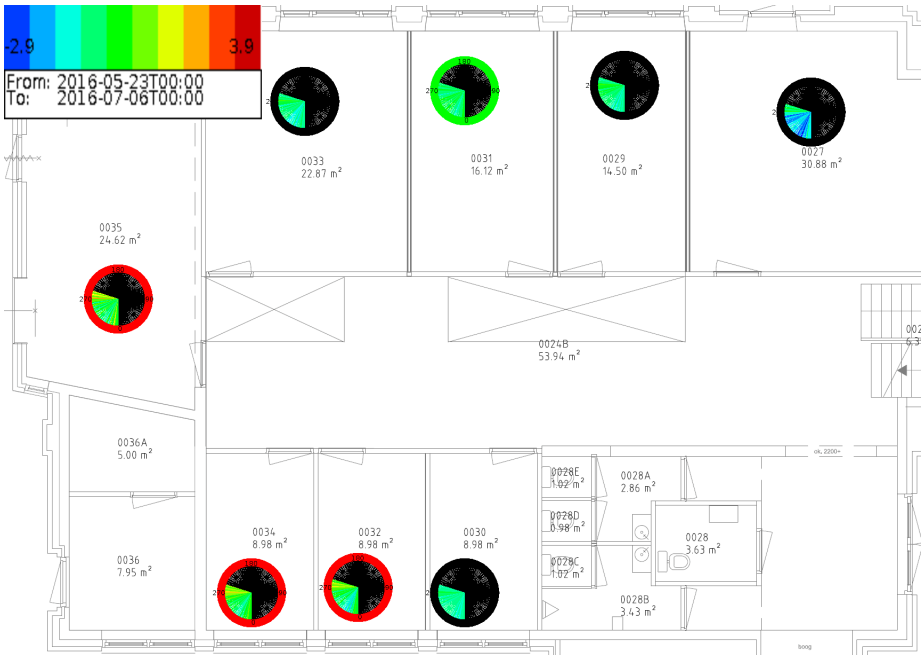


Figure 5.8: Clustering of some offices, using the prediction errors.

5.2.4 Clustering Model Parameters

Figure 5.9 displays the clustering as produced by the model parameter technique. This is where the offices are clustered using the room temperature model parameters as the features. It is the same area of the building as Figures 5.7 and 5.8. The first thing worth mentioning is the fact that the inner part of the circles are almost entirely black. This is because the number of model parameters is much smaller than the number of lines that are drawn for the circle, 360. Because the time period is the same as for the other figures, there is only one colored line drawn in the circle. This is fine, seeing as plotting the model parameters in this way does not make much sense. It would be much harder to interpret than the time series data, because there is less data, and because the data points are not related to each other, preventing any pattern from being meaningful.

The clustering in Figure 5.9 is a fair bit different than the ones in Figures 5.7 and 5.8. This is true for most of the building, there are some areas where the clustering is identical, but for most areas it differs. This can be explained by the fact that the difference in quality between the room temperature models is quite significant. Section 5.3 will elaborate on the quality of the room temperature models.

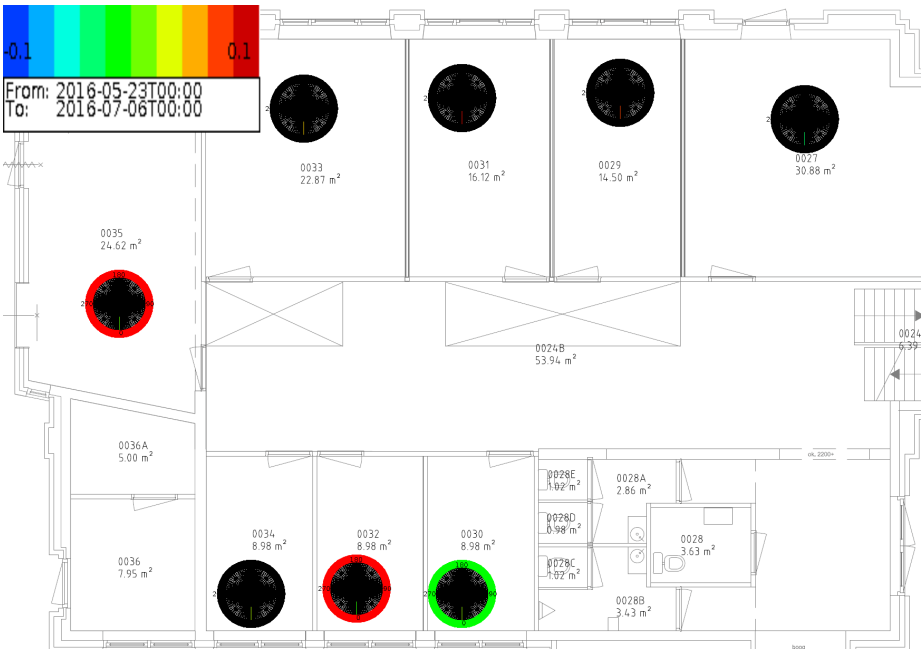


Figure 5.9: Clustering of some offices, using the model parameters.

5.2.5 Similarity of Clusterings

One way of assessing how similar two clusterings are is by using a form of the Hamming distance. Given two string, the Hamming distance is the number of

	Room Temperature	Prediction Error	Model Prediction
Room Temperature	1.0	0.8	0.56
Prediction Error		1.0	0.61
Model Prediction			1.0

Table 5.1: Similarity scores between the clusterings. Zero indicates that the clusterings are completely dissimilar, one means the clusterings are identical.

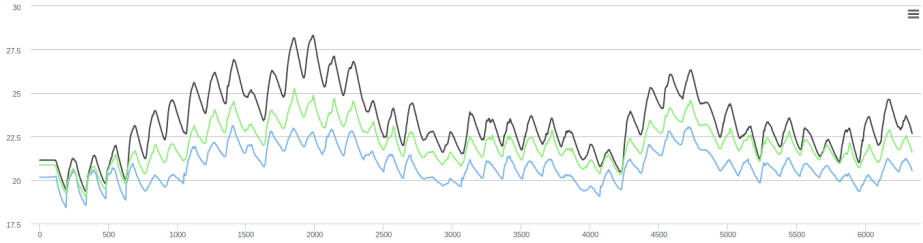
positions where the corresponding characters are unequal. We can view the clustering as a string: each office belongs to one of the clusters, which we can label. Normalizing this distance by dividing by the number of offices gives some insight into the similarity of the clusterings. A score of zero means the two clusterings are identical, while a score of one indicates that the clusterings are completely dissimilar. We then subtract this score from 1, to obtain a more intuitive similarity score, where 1 means they are identical, and 0 means they are completely different.

One difficulty in determining the similarity between clusterings is the fact that the labels likely do not correspond between clusterings. Moreover, when using different clustering techniques there is really no way to match cluster centers to each other in order to get a consistent labeling. Figure 5.10 displays the cluster centers of the three different clustering techniques to illustrate this point. There is no way to tell what cluster center belongs to any other cluster center of another technique.

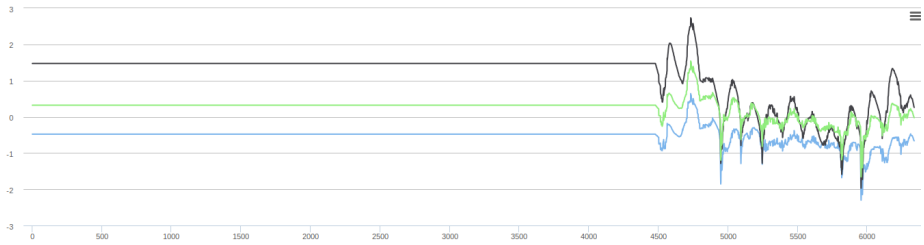
In order to evaluate the similarity of clusterings it is imperative that we can somehow match clusters from different techniques to each other. We deal with this by computing the similarity for each permutation of the matching between clusterings, and then taking the highest similarity as the right one. This may sound scary, like tweaking the results, but from our analysis it is actually legitimate to do this. Say the clusters are labeled 1, 2 or 3. Then one clustering might look like this: $1, 1, 0, 2$, and the clustering from another technique might look like this: $2, 2, 1, 0$. These clusterings are actually identical, the only difference is the labeling.

Table 5.1 contains the results from the comparison of how similar the clusterings of the different techniques are. The numbers on the diagonal are all one, as here the clusterings are compared to themselves. More interesting are the numbers in the upper right corner. The most similar clusterings are achieved by the room temperature and the prediction error techniques, at 80% similarity. The second highest similarity comes from the comparison between the prediction error and the model prediction techniques, at 0.61% similarity. The lowest similarity results from the comparison between the room temperature and the model prediction techniques, at 0.56%.

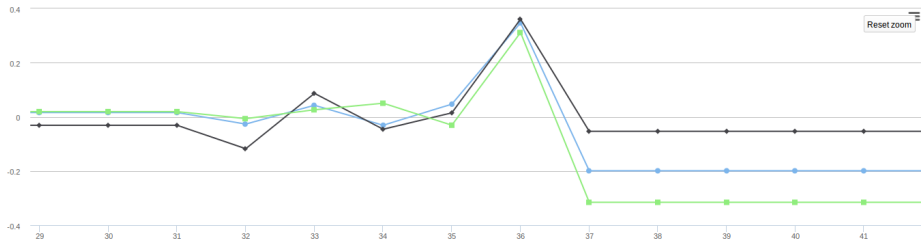
These numbers inspire confidence in the fact that the produced clusterings are meaningful, particularly the clusterings produced by the room temperature and the prediction error techniques. Two independent methods that produce clusterings that are 80% similar shows that there is in fact likely an underlying factor that is responsible for these clusters. Section 5.4 expands more on the feature engineering part.



(a)



(b)



(c)

Figure 5.10: Cluster centers of (a) the room temperature clusters, (b) the prediction error clusters, and (c) the model prediction clusters.

5.3 Room Temperature Model

The quality of the room temperature models is evaluated using two different measures of model fitness: R-squared (R^2) and the Root Mean Square Error (RMSE). The R^2 value gives an indication of the proportion of the variance in the predicted variable that is explained by the model, all else being equal, a model with a higher value is generally better than a model with a lower value. The RMSE is a measure of the differences between the values as predicted by the model, and the ground truth. Here the opposite is true from the R^2 value, now a lower value is generally better than a higher value.

The data is split up in two parts: a training set and a test set. The training set is used to create the model, after which the model quality is determined using the test set. This is done by using the created model to predict the values over the test set range, and then comparing these predictions to the ground truth of the test set. These values are then used to compute the R^2 and RMSE metrics.

Before the whole building was equipped with sensors and actuators, we deployed some in one office for testing. Here we collected some data from the actuator, but there was only about a month where we were able to measure the radiator temperature before the heating system was turned off. Figure 5.11 shows the predictions of the model that was trained with a training set consisting of a subset, 80% of the data, of the values from this month, and evaluated with a test set that consisted of 20% of the data. Both the ground truth, labeled "actual" in the figure, as the predicted values are plotted. The R^2 score of this model is 0.50, with an RMSE of 0.12. The features that are used to create this model are:

- Valve temperature
- Outside temperature
- Weather classification
- KNMI sun values [15]
- Bias
- Weekdays dummy variable
- Weekends dummy variable
- Moving average of valve temperature (N=5)

The moving average of the valve temperature was added in an attempt to smooth the values of the valve temperatures, seeing as they showed some peaks that influenced the prediction. The KNMI sun values are taken from a different weather station than the other weather data, at the local airport Eelde. While useful in this analysis, it is excluded from the design of the model creation because the data is not available in real time, it can be requested in batches. Because it is important that the heating system is controlled in real-time, this feature was excluded for that part.

The figure shows that the two lines mostly follow the same general trend. There are two exceptions, at around data point 390 and point 580, the actual tem-

perature was lower than the predicted value. This is possibly due to some unexpectedly cold nights.

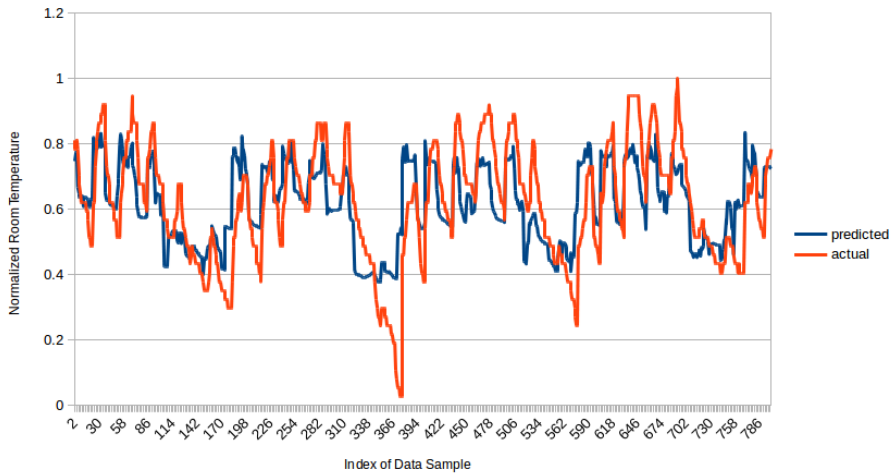


Figure 5.11: Predicted values and the ground truth for one office in the Nieuwenhuis building, about four weeks.

Aside for the modeling of the one office for which we have the valve temperature data, we also created models for all offices without the valve temperature. Figure 5.12 shows the R^2 and RMSE values for each office. The data used for this modeling was collected over the period of a month. The training set was 80% of the data, and the testing set was 20%. The labels on the x-axis are the MAC addresses of the room sensors, only a subset is shown as not all labels could fit. The difference between offices is quite significant: the office with the best model has an R^2 of about 0.5, while the office with the lowest quality model has a score of about 0.07. The average model score seems to be about 0.3. The values of the RMSE show less variance between offices. For the office with an R^2 of about 0.07, the RMSE is close to 1.0. The average RMSE is about 0.85.

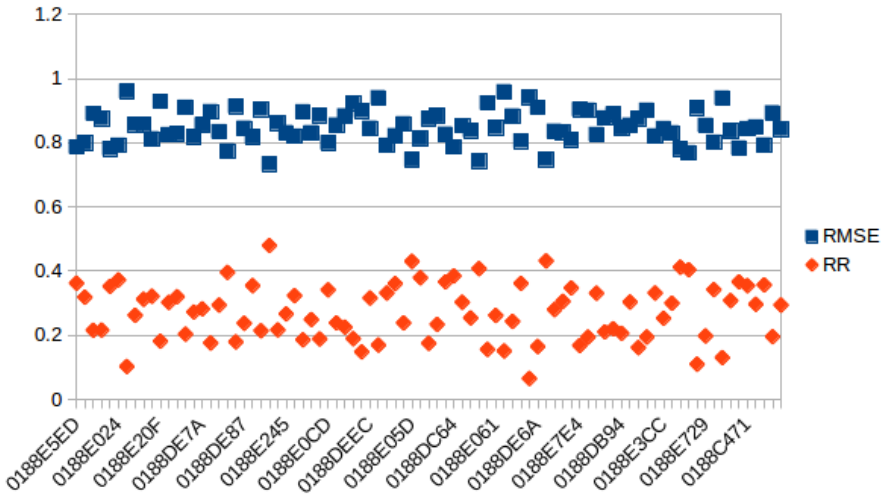


Figure 5.12: R^2 and RMSE scores for all offices. Only a subset of MAC addresses is displayed on the x-axis.

5.4 Feature Engineering

In order to evaluate the ability to discover new features, we apply a form of cross-validation to the clustering of the model predictions. We choose to use the model predictions because here individual models are used for the offices, so the effect that leaving out one of the features will have will be more prominent than when one big model is used for all the offices. One at a time the following features are left out: presence, weather classification and the outside temperatures.

	Model Prediction Clustering
Clustering with weather classification left out	0.44
Clustering with presence left out	0.75
Clustering with outdoor temperature left out	0.84

Table 5.2: Model prediction clustering compared to model prediction clustering with different features left out.

Table 5.2 displays the similarity scores of the clusterings with different features left out, compared to the original model prediction clustering. The similarity of the clustering with the outdoor temperature left out is the greatest, indicating that the outdoor temperature might have a smaller effect on the model than the other two features. The similarity of the clustering with the weather classification left out is the smallest, seeing as this is mostly a feature that represents the sun, this could mean that the sunshine has the largest effect of the three features. In the following three sections we assess the features that were left out individually, as we are interested in figuring out whether we could have discovered them had they not initially been part of the model.

Figure 5.13 shows the entire clustering of the model parameters. The three floor levels are separated by the black horizontal lines, and each floor level is indicated to the left of the figure. This figure serves as a reference to compare the next figures to, so we can compare the effect that leaving out each of the respective parameters has on the clustering.

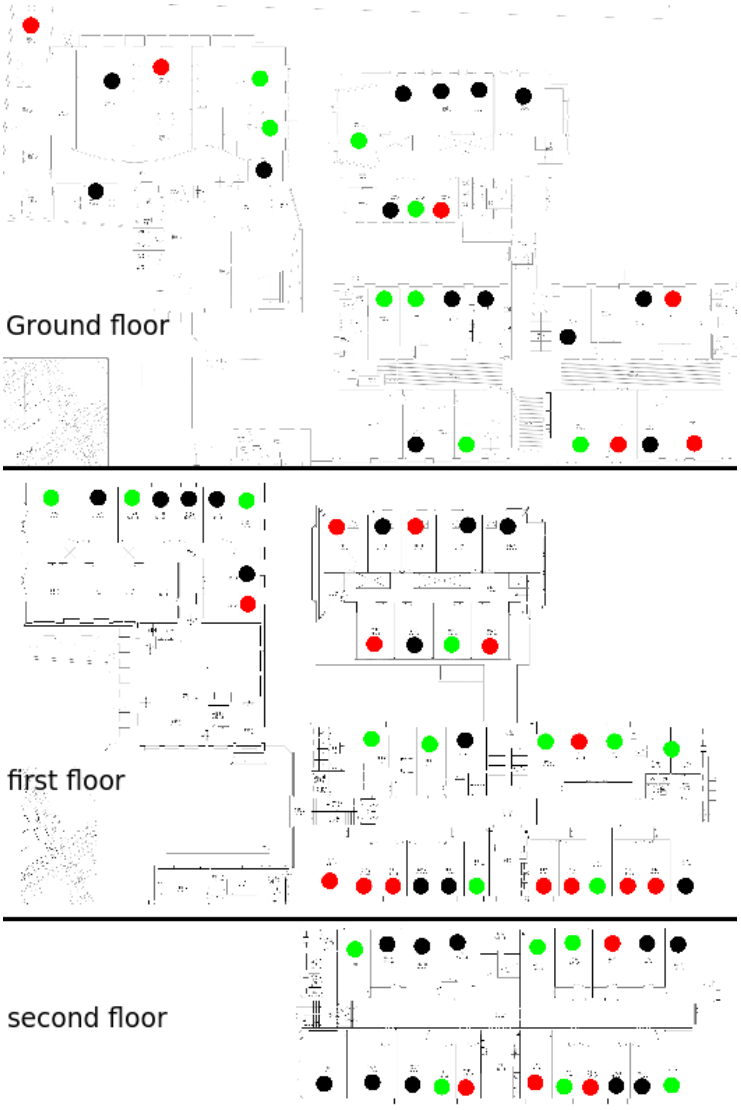


Figure 5.13: Clustering of the model parameters. Floors are separated by the horizontal black lines.

5.4.1 Leaving out the weather classification

Figure 5.14 shows the clustering of the model parameters with the weather classification feature left out. The north is indicated by the arrow on the top-

right of the figure. Knowing where the north is might be relevant, as we are investigating the role of the sun on the clustering. It seems reasonable to expect the side of the building that gets the most sun to be grouped together in a cluster. Of course, seeing as the building is surrounded by other buildings and objects such as trees, this may not be as straightforward.

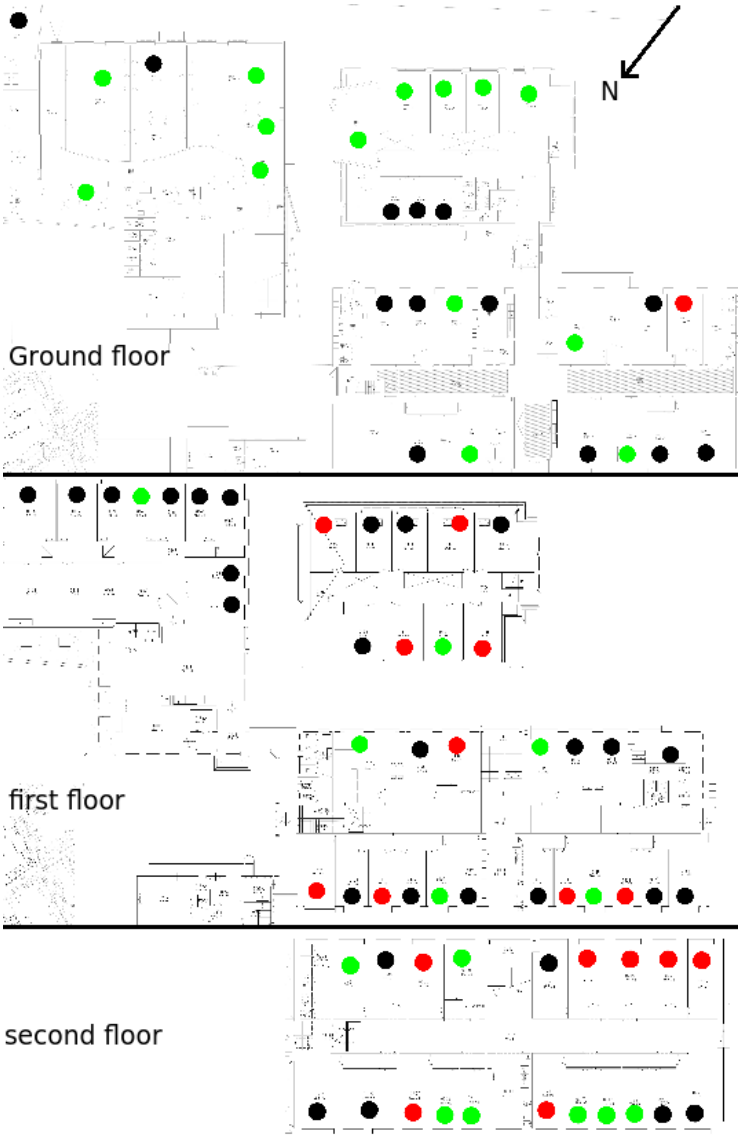


Figure 5.14: Clustering with the weather classification left out as the model features. Floor levels are separated by the horizontal black lines. The north is indicated with the arrow in the top-right corner.

Figure 5.14 is the figure we would have in case we had not yet thought of the weather classification, i.e. is it cloudy, sunny or raining, as a potential feature for the model. The question now is whether we can somehow, by comparing

this figure to Figure 5.13, trace back the steps that could have led us to discover the weather classification as a feature. We start by looking for anything that stands out in Figure 5.14, and see if it can be explained by the missing feature.

The first thing we notice is the underrepresentation of the red cluster on the ground floor. Only one office on the ground floor is assigned to this cluster, while on the first and second floor many more floors are assigned to the red cluster. A number of explanations come to mind that could explain a difference between the ground floor and the higher floors. We might for example consider the fact that the ground floor is, more or less by definition, directly connected to the ground, i.e. the earth. It could be possible that the temperature or thermal conductivity of the earth beneath the building could play a role. This is not particularly visible in Figure 5.13, here the distribution of clusters is more balanced.

Another explanation for the difference between the ground floor and the higher floors could be the sun. Particularly if the building is surrounded by other buildings or trees, this could block the sunshine from entering the offices on the ground floor, but would permit it to enter the offices on the higher floors. If the sun does have an effect on the office temperature, it is reasonable to also see a difference in offices based on their orientation, towards the north or south. This is not very obviously present in the figure, but one piece of information does indicate that it might be the case to some degree. The fact that the green cluster is very prominent on the ground floor, as well as on the north corner of the second floor, might be an indication that they share the same underlying feature, in this case the sun.

5.4.2 Leaving out the presence

Figure 5.15 shows the clustering of the model parameters with the presence left out as a feature. Attempting to discover presence as a feature based only on this figure is going to be very hard, as the presence of an office is specific to that office. Without any knowledge about the actual presence of each office, coming up with presence as a feature based only on this figure is very unlikely. For example, we can see that the offices assigned to the red cluster are in the minority on all floors. In fact, out of about 85 offices, only six are assigned to the red cluster. Without any very specific domain knowledge, in this case information that may lead the modeler to observe a link between the offices in the red cluster, we have to conclude that deriving presence from this figure is unlikely to happen. This finding is somewhat backed up by the fact that similarity of the clustering without presence compared to the original model parameter clustering is higher than the similarity of the clustering with the weather classification left out, as showed in Table 5.2. This indicates that the presence plays a less important role than the weather classification.

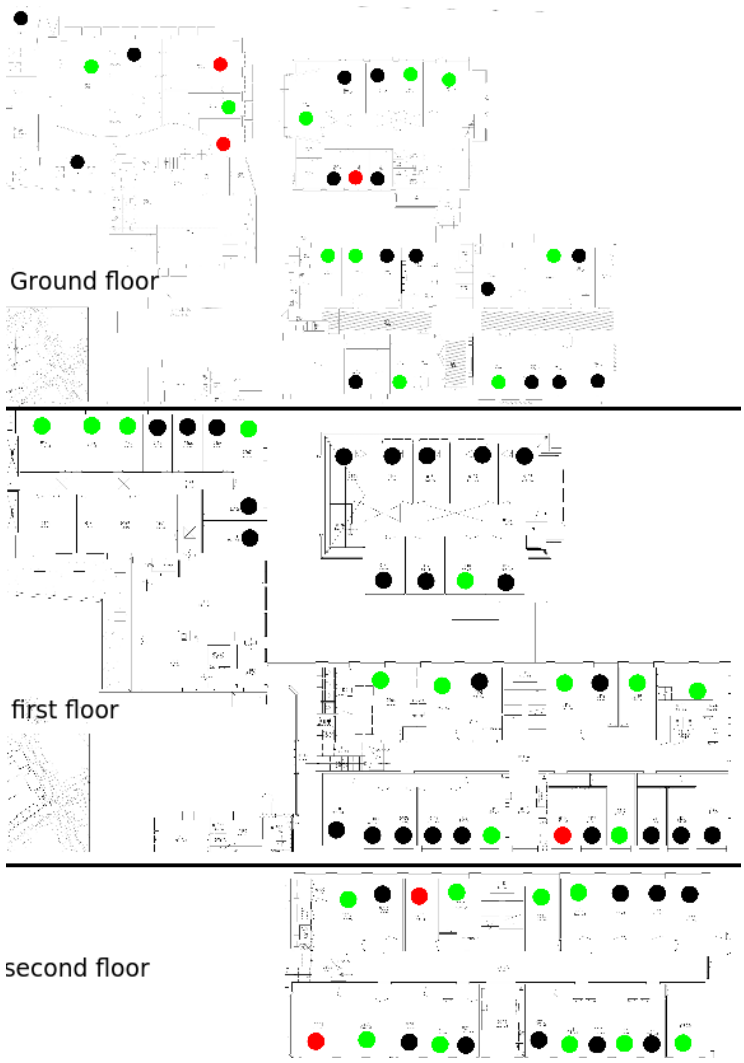


Figure 5.15: Clustering with the presence left out as the model feature.

5.4.3 Leaving out the outdoor temperature

Figure 5.16 shows the clustering with the outdoor temperature feature left out from the models. We notice that offices assigned to the green cluster are less well represented on the ground floor and second floor, while on the first floor the distribution of clusters is quite even. Black seems to be the most prevalent cluster on all floors. On the ground floor we notice two green offices in the bottom right corner of the building. This could have been explained by something like a cold or warm flow of wind in the street, however then the green offices are also prevalent north side of the first floor of the main building and the patio on the first floor. Linking this to the outdoor temperature based on only this figure would be a bit of a stretch. However, perhaps linking it to wind direction and speed, it is not entirely unthinkable that a particular flow of outside air affects

the temperature of certain offices.

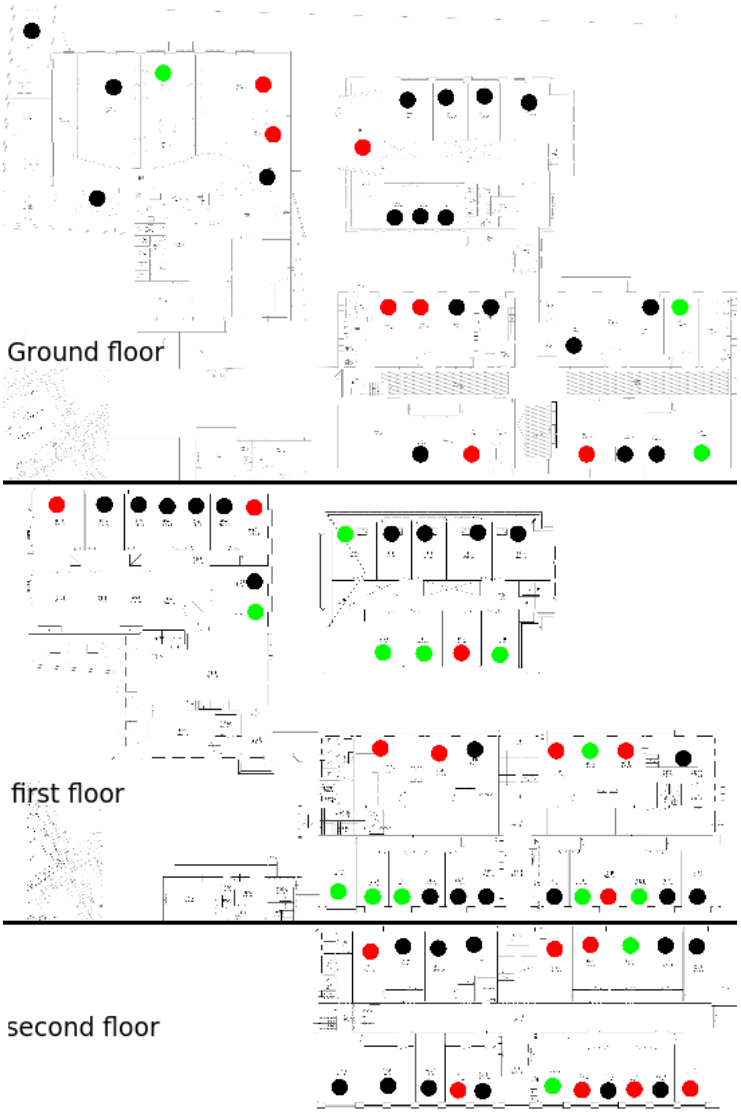


Figure 5.16: Clustering with the outdoor temperature left out as the model feature.

5.4.4 Conclusion

The technique to discover new features as presented above is very much a subjective process. It should be seen more as a tool that aids in discovering features rather than a precisely defined set of steps that will guarantee some result. Reasoning backwards like we have done here is relatively easy, as we know what we are looking for. Had this not been the case, the question of whether or not the features would be discovered remains. However, this does provide at least some

guidance as to where, and how, to look for them.

5.5 Discussion

The research questions, as originally posed in Chapter 1, are:

1. How can we discover new features that can be used for improving the data models?
2. How can context be derived from a smart heating system?
3. How can the negative effect of a smart heating system be minimized, based on the gathered context information? i.e. how can we control the radiators in such a way that we optimize the trade-off between user satisfaction and saving energy?

In this chapter we look at to what extend these questions have been answered, what problems arose from answering them and what future directions this work can go in.

The feature engineering method as proposed in this work serves more as a set of guidelines in discovering features than an exact set of steps that will guarantee some result. The process relies heavily on the domain: for certain domains it will not work at all. The reason for this is that it is not always possible to visualize the data in a way that will aid the process of discovering features. In our case, where we applied the process to a smart heating situation, we have the benefit of dealing with a real-world phenomenon that has a natural spatial distribution, i.e. the offices have a location and some orientation to each other. For more abstract types of data the visualizations used here may not work at all, and thus the value that this feature engineering process offers may be limited in this case.

However, as we have seen in this chapter, if applied in the right manner and to the right conditions, the feature engineering method is capable of aiding the data modeler in discovering new features. Not because it is guaranteed to produce new features given some input, but because it will facilitate and stimulate the thought process of the data modeler. The feature engineering process is ideally carried out in collaboration with a domain expert, because when questions arise as a result from studying the visualization and the clustering, they are likely able to better connect the dots given their domain knowledge. They may also be the ones coming up with the questions in the first place. It is these questions, resulting from studying the data visualization and clustering, that eventually may lead to the discovery of new features. An example in this case is the discovery of the ground temperature and ground conductivity. These are not features that we thought of when we first created the model, but after studying the clustering they are one explanation for why the offices on the ground floor behave differently from the ones on higher floors.

The second research question mostly pertains to the modeling of the office temperature. Even though the office temperature is only part of the entire context of the environment, it is the most relevant one for the case of smart heating.

Another way to phrase the question is: were we able to derive the office temperature based on the data we had available? The best models we created had an R^2 value in the 50% range. Whether or not some R^2 value is good depends entirely on the specific situation, in some cases a lower value can be acceptable, while in another a very high R^2 value is required. In this particular scenario an R^2 value in the 50% range is pretty good, especially if we consider the fact that one of the most important features, radiator temperature, was missing in the creation of these models. This is backed up by Figure 5.11, where we can see that the predictions correspond quite well to the actual values. There were also a lot of models with a (much) lower R^2 value, down to below 10%. Compared to the highest models the ones with the lower scores will translate into a worse performance of the smart heating system. Part of the future work will be figuring out why the models are so much worse for some offices compared to others, and seeing if we can somehow account for that in the control part.

For the third research question we have proposed a design for the smart heating system. There are some risks associated with the design. For example, the success of the system is largely determined by the quality of the room temperature model, and as we have seen this quality does differ quite significantly between the offices. The risk here is that for those offices with a lower-quality model, the temperature may not always be at an ideal level. Another risk lies in the prediction of the occupant's presence: it will depend largely on the individual how regular their presence in the office is. Some people may exhibit very regular behavior: arrive at the same time, take breaks at the same time and leave at the same time, while others may have irregular meetings and other factors that break their presence pattern. Thus, an accurate prediction of the occupant's presence is paramount to the success of the smart heating system. It is worth considering some form of "presence accuracy detection", i.e. a measure of how accurately we can predict the presence of an office, and then use this to set the minimum temperature set point. What this allows for is a lower chance of making the occupant uncomfortable, because we can heat up the room quicker as the temperature was higher to begin with. Applying this to offices that we have difficulty with predicting the occupant's presence of can improve the comfort of users, at a cost of resulting in fewer savings.

Chapter 6

Conclusion

In this work we presented a method to discover new features for linear regression models. The method consists of two major approaches: visualizing the data and clustering the data. This is done in order to facilitate and stimulate the thought process of the data modeler: providing visual inputs that will spark questions about the model, the data and the entity being modeled in its environment. These questions are the stepping stone to discovering new features, as the process of answering them will reveal potential features that may not have been initially considered.

The feature engineering method is evaluated using a smart heating system. In a deployment of about 100 offices this system attempts to find the best solution to the trade-off of the occupant's comfort versus the amount of energy saved. We presented a design and deployment of this system, however were unfortunately unable to evaluate its performance due to the hot summer weather, causing the heating system to have been turned off. The application of the feature engineering method in a smart heating context has resulted in the discovery of at least two potential features: the temperature of the ground below a building, and the conductivity of the building with the earth.

6.1 Future Work

There are a number of directions that any potential future work can take. One of the next steps could be applying this method in a different setting. Here we chose to go for a smart heating system, but evaluating this in another setting would greatly improve the credibility of the proposed method. Also, incorporating the newly discovered features into the model and then evaluating whether or not the model has actually improved is also a potential next step.

In the specific case of the smart heating system there a number of things one can look at. As mentioned before, the presence prediction is very important for the success of the system. For this work we did not go into detail about this, due to having to scope the project. Future work can consist of assessing viable methods for predicting the occupant's presence. One can also consider adding

more features to the model, such as the UV-strength of the sun, or the ground temperature.

One question that is still left unanswered is why the offices exhibit such large differences in the quality of the models. At this point it is still unclear, and this might actually also function as an opportunity to discover new features: possibly there is an underlying factor that causes these differences in model quality.

The data we have collected can also be used for other purposes than just feature engineering or smart heating. For example, it may be used for "event detection": finding unusual changes in the data that can be explained by some event, for example a "bring your kid to work day", or a festive activity at the office. Building managers could use this data to gain insight into the occupational state of the offices: they might be interested in knowing when an office has been empty for a long time, so they can investigate what is going on. In case of flexible working places it could be used to optimize the planning of the occupation of these work places.

Acknowledgements

Thanks to the SustainableBuildings company and the surrounding team members, for providing the tools, environment and support necessary to deploy the smart heating solution and carry out this project. Thanks to Peter Hartman and Ruben van der Leij of the university's facility management department for their support throughout the entire project, and helping with the installation of the smart heating solution. I would like to thank Michael Biehl for the discussion about different clustering techniques. Finally I would like to thank my supervisors for their guidance during the project.

Bibliography

[1] Grote Rozenstraat 38 – Nieuwenhuis building . http://www.rug.nl/gmw/organization/profile/addresses/buildings/grrozenstraat38?lang=en, 2016. [Online; accessed 15-June-2016].

[2] Jenkins - Build great things at any scale. https://jenkins.io/, 2016. [Online; accessed 3-August-2016].

[3] Apache Spark - Lightning-fast cluster computing. http://spark.apache.org/, 2016. [Online; accessed 3-August-2016].

[4] Apache Spark MLlib. http://spark.apache.org/mllib/, 2016. [Online; accessed 1-August-2016].

[5] Determining the optimal number of clusters: 3 must known methods - Unsupervised Machine Learning. http://www.sthda.com/english/wiki/determining-the-optimal-number-of-clusters-3-must-known-methods-unsupervised-machine-learning, 2016. [Online; accessed 28-July-2016].

[6] Docker. https://www.docker.com/what-docker, 2016. [Online; accessed 3-August-2016].

[7] DolphinView Advanced. https://www.enocean.com/en/download/, 2016. [Online; accessed 16-July-2016].

[8] Energy Harvesting Wireless Sensor Solutions and Networks from EnOcean. https://www.enocean.com/en/, 2016. [Online; accessed 16-July-2016].

[9] EnergyPlus. https://energyplus.net/, 2016. [Online; accessed 12-July-2016].

[10] Enocean USB gateway. https://www.enocean.com/en/enocean_modules/usb-300-oem/, 2016. [Online; accessed 29-June-2016].

[11] Gebouweninformatie. http://www.rug.nl/gmw/organization/profile/addresses/buildings/, 2016. [Online; accessed 15-June-2016].

[12] Git. https://git-scm.com/, 2016. [Online; accessed 4-August-2016].

[13] GitHub. https://github.com/, 2016. [Online; accessed 4-August-2016].

[14] Kieback and Peter. http://www.kieback-peter.de/de-en/, 2016. [Online; accessed 29-June-2016].

[15] KNMI - Data ophalen vanuit een script. http://www.knmi.nl/kennis-en-datacentrum/achtergrond/data-ophalen-vanuit-een-script, 2016. [Online; accessed 18-August-2016].

[16] Linear regression. https://en.wikipedia.org/wiki/Linear_regression#/media/File:Linear_regression.svg, 2016. [Online; accessed 31-August-2016].

[17] RabbitMQ. https://www.rabbitmq.com/, 2016. [Online; accessed 18-July-2016].

[18] Raspberry Pi foundation. https://www.raspberrypi.org/, 2016. [Online; accessed 29-June-2016].

[19] RFID 4122 - A Universally Unique IDentifier (UUID) URN Namespace. https://www.ietf.org/rfc/rfc4122.txt, 2016. [Online; accessed 4-August-2016].

[20] The Apache Cassandra Project. http://cassandra.apache.org/, 2016. [Online; accessed 18-July-2016].

[21] The Scala Programming Language. www.scala-lang.org/, 2016. [Online; accessed 4-August-2016].

[22] Using etcd with CoreOS. https://coreos.com/etcd/, 2016. [Online; accessed 3-August-2016].

[23] Weave Net - Weaving Containers into Applications. https://github.com/weaveworks/weave, 2016. [Online; accessed 3-August-2016].

[24] R. ACS. Bayesian classification (autoclass): Theory and results. 1996.

[25] D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, 1994.

[26] J. C. Bezdek. *Pattern recognition with fuzzy objective function algorithms*. Springer Science & Business Media, 2013.

[27] G. A. Carpenter and S. Grossberg. A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer vision, graphics, and image processing*, 37(1):54–115, 1987.

[28] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[29] S. Garavaglia and A. Sharma. A smart guide to dummy variables: Four applications and a macro. In *Proceedings of the Northeast SAS Users Group Conference*, page 43, 1998.

[30] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[31] J. Han, J. Pei, and M. Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[32] L. Kaufman and P. J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009.

[33] E. J. Knibbe. Building management system, Oct. 15 1996. US Patent 5,565,855.

[34] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[35] R. Krishnapuram, A. Joshi, O. Nasraoui, and L. Yi. Low-complexity fuzzy relational clustering algorithms for web mining. *IEEE transactions on Fuzzy Systems*, 9(4):595–607, 2001.

[36] T. Lai, H. Robbins, and C. Wei. Strong consistency of least squares estimates in multiple regression ii. *Journal of Multivariate Analysis*, 9(3):343 – 361, 1979.

[37] L. Lan, Z. Lian, and L. Pan. The effects of air temperature on office workers' well-being, workload and productivity-evaluated with subjective ratings. *Applied ergonomics*, 42(1):29–36, 2010.

[38] B. Lehmann, V. Dorer, and M. Koschenz. Application range of thermally activated building systems tabs. *Energy and Buildings*, 39(5):593 – 598, 2007.

[39] T. W. Liao. Clustering of time series data—a survey. *Pattern recognition*, 38(11):1857–1874, 2005.

[40] Z. Liao and A. Dexter. A simplified physical model for estimating the average air temperature in multi-zone heating systems. *Building and environment*, 39(9):1013–1022, 2004.

[41] J. Lu, T. Sookoor, V. Srinivasan, G. Gao, B. Holben, J. Stankovic, E. Field, and K. Whitehouse. The smart thermostat: Using occupancy sensors to save energy in homes. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, SenSys '10, pages 211–224, New York, NY, USA, 2010. ACM.

[42] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967.

[43] J. B. Rawlings and D. Q. Mayne. *Model predictive control: Theory and design*. Nob Hill Pub., 2009.

[44] G. Ríos-Moreno, M. Trejo-Perea, R. Castañeda-Miranda, V. Hernández-Guzmán, and G. Herrera-Ruiz. Modelling temperature in intelligent buildings by means of autoregressive models. *Automation in Construction*, 16(5):713–722, 2007.

[45] Y. Saeys, I. Inza, and P. Larrañaga. A review of feature selection techniques in bioinformatics. *bioinformatics*, 23(19):2507–2517, 2007.

[46] J. Scott, A. Bernheim Brush, J. Krumm, B. Meyers, M. Hazas, S. Hodges, and N. Villar. Preheat: Controlling home heating using occupancy prediction. In *Proceedings of the 13th International Conference on Ubiqui-*

tous Computing, UbiComp '11, pages 281–290, New York, NY, USA, 2011. ACM.

[47] O. Seppanen, W. J. Fisk, and Q. Lei. Room temperature and productivity in office work. *Lawrence Berkeley National Laboratory*, 2006.

[48] D. Sturzenegger, D. Gyalistras, M. Gwerder, C. Sagerschnig, M. Morari, and R. S. Smith. Model predictive control of a swiss office building. In *Clima-RHEVA World Congress*, pages 3227–3236, 2013.

[49] R. L. Thorndike. Who belongs in the family. *Psychometrika*, pages 267–276, 1953.

[50] W. Wang, J. Yang, and R. Muntz. Sting: A statistical information grid approach to spatial data mining. In *VLDB*, volume 97, pages 186–195, 1997.