B.Sc. Thesis
Bachelor of Science in Mathematics and Technology

DTU Compute
Department of Applied Mathematics and Computer Science

Remote assessment of buildings thermal energy performance

In cooperation with SEAS-NVE

Salina Lam (s144238)
Sebastian Balle (s144243)

Kongens Lyngby 2017

DTU Compute
Department of Applied Mathematics and Computer Science
Technical University of Denmark

Matematiktorvet
Building 303B
2800 Kongens Lyngby, Denmark
Phone +45 4525 3031
compute@compute.dtu.dk
www.compute.dtu.dk

Abstract

This report has its inspiration from the app WATTS created by SEAS-NVE. Thermal performance using heat load has been analysed before writing this report, but measures of electricity have not yet been examined. The need to know our consumption patterns is something that increasingly interests consumers. The analysis has been conducted in the statistical program `R`.

This report will deal with a certain amount of Danish houses and their electric consumption. The purpose of this paper is to propose a possible expansion of the WATTS app. This paper will also provide SEAS-NVE with methods which can help them group house into different performance groups.

Throughout the report linear regression, classification and clustering methods will be used. The methods applied in this report are used to display different angles of the performance of the house.

From the results, we see that we were able to build a model which was unique for each house. It was possible to cluster these houses based on their estimates in the regression model. In the classification part, we were able to obtain an accuracy of 58.8 % when predicting future energy labels. SEAS-NVE can use this report to detect vulnerable customers and advise them regarding the performance of their house. The findings in this report are specific for the houses in our data set but can be expanded to other houses. Future extension to the analysis has been proposed to get a more thorough analysis.

Preface

This thesis was written at DTU Compute to complete the BSc programme in Mathematics and Technology. It is written by two people in collaboration with the company SEAS-NVE. This thesis is rated to 20 ECTS points and written during the period February 1st 2017 to June 23rd 2017.

We would like to give a thanks to our supervisor Peder Bacher for advisory, encouragement and feedback throughout the process. Also, thanks to Anders Spur Hansen and Jon Liisberg for helping with the data retrievement and giving us ideas.

The project was designed and written with inspiration from the given project description:

In this project the possibilities for estimating the thermal energy performance of buildings with electrical heating systems will be studied. From several studies we know, that the thermal performance can be estimated from measurements of heat load, however it has not been studied how this can be done from measurements of electricity. The research hypothesis is, that linear regression models can be applied to daily values of electricity consumption and provide estimates which can be used to energy label the building. The data used will be hourly and daily electricity readings coupled with climate model data, and further information available about the building and system will be included. Naturally, the building need to be heated with an electrical driven heat source, either direct electrical heating or some type of heat pump, and the possibilities for separating the building and the heating system will be investigated, as well as the influence of occupancy behaviour on the uncertainty of the estimates. All this will lead to statistical models which can provide useful information which can be presented in a simple way to the building users on how their building performs.

Contents

CHAPTER 1

Introduction

1.1 Intro to WATTS app

WATTS is an app created by the Danish energy supplier SEAS-NVE. The app provides several useful applications such as forecasting electricity consumption and keeping track of the budget. The user is also able to compare their consumption with similar users. The applications appeal to many different users. The customer can see when the electricity is environment-friendly, the owner of the solar cell can see their daily profit and the ordinary family can plan their budget. WATTS can give you a straightforward and quick overview of basic things you need to know about your electricity consumption. It is possible to develop the app further, by offering users a guiding energy label of their house and simplify the understanding of a houses performance.
WATTS is currently only available for SEAS-NVE energy customers with a remotely read electricity meter.

1.2 Energy labelling

In Denmark, it is compulsory to get the energy label of a house concerning sales or rental. The label is only valid for seven years, assuming no remodeling of the building is done. An energy label is a measure of a building's energy efficiency and often affects sales of a house. Most of the buildings in Denmark carry a label between C and G, so often a status rapport is necessary to improve the energy efficiency and increase the chance of sale and increase the sales prices. This can be a very costly affair with prices going up to 7.113 DKK for buildings below 300 m^2. The prices quickly rise if the status report with suggestions for improvements is included as well. In practice, energy labels are measured by different factors mainly the calculated and measured power consumption. Also, an examination of the specifications of the building such as heating type and isolation are taken into consideration when labeling a house. The examination is performed by certified energy consultants.

1.3 Motivation

The aim of this report is to show the possibilities of offering users of SEAS-NVEs WATTS app a guiding data-driven free energy label mainly based on their consumption and consumption patterns. The data-driven labels will also be based on meteorological variables and self-entered data. Users will be able to get an idea of the pitfalls of their homes and where improvements can be made to bring down their electricity bill. Usually, a certified energy label is only ordered when selling or renting a house, because it can be very pricey. Through this way, we can offer a mini status report to any user, where the user can decide whether they want to follow the advice or not.

Our approach to energy labeling will be different from the certified way as we will use statistical analysis and machine learning to classify a house based on other variables than the specifications of a house. The benefits of this approach are that if you do remodel your house, or make drastic changes in your consumption pattern, these methods should be able to detect a change, thus change your energy label accordingly. Of course, this requires sufficient data, so it can take many days to see the changes.
It should be noted that our labels are only guiding and that it can not replace a certified energy label, but it can offer guidance and insight to the user and SEAS-NVE as well. Precaution must be taken as there can be many unexplained phenomena that determine the advice and labeling given. This could be different preferences in the baseline indoor temperature or whether or not a house is sheltered. [10][16].

CHAPTER 2

Data

Data was provided by SEAS-NVE in two separate CSV-format files. The main dataset contained 29 attributes and 574,670 data points. The dataset included explanatory weather variables, energy production/sale, and dates. The second data set included additional data from "Bygnings og Boligregistret" (BBR) including official energy labeling of each building. This set included 87 observations with 16 attributes. The attributes `Residents` and `SquareMeters` were entered by the resident of the house. All production/sale data was measured by SEAS-NVEs own AMR monitors. Weather data was measured by ECMWF, the European Center for Medium-Range Weather Forecast.

For the further analysis, we will distinguish between a regression and classification data set, made from the two original datasets. This section will mainly focus on how the regression data was obtained. In Section 5.1 it will be discussed what and why the classification data set is different from the regression.

2.1 Original data

The original data included hourly observations from the period 31-12-2015 to 31-12-2016. The following table shows the attributes from the BBR data set.

Variable	Description
City	City name HeatingType
Type of heating - entered by the user	
HouseType	Type of house - entered by the user
HeightOverSeaLevel	Height above sea level
Residents	Number of residents in house
SquareMeters	Size of house
BBR_ApplicationCode	Type of building
BBR_YearOfConstruction	Construction year
BBR_HeatingType	Type of heating - according to BBR
BBR_HouseType	Type of house - according to BBR
CurrentEnergyLabel	Energy label
PossibleEnergyLabel	Proposed Energy Label

Table 2.1: Attributes from the original BBR data set.

The other dataset contained the following attributes.

Variable	Description
V1	Incremental number
SK_Household	18 digit unique id for each house
StartTime	StartTime for measure
EndTime	EndTime for measure
EnergyPurchase	Purchased electricity in between StartTime and EndTime
EnergySale	Sold electricity in between StartTime and EndTime
DefCode	Type of building
Lattitude	Approximately lattitude of buildings position
Longitude	Approximately longitude of buildings position
AirTemperature	Outside airtemperature
AtmosphericPressure	Atmospheric pressure outside
SurfaceSolarRadiationDownwards	Total solarradiation recieved from the sun
TotalSkyDirectSolarRadiationAtSurface	Total solarradiation recieved perpendicualar from the sun
WindSpeed	Outside windspeed at 10 m
WindDirectionDegree	Wind direction in degrees at 10 m
WeekDay	Which day of the week in number (i.e. Monday = 1)
IsWeekDay	Displays 1 if date is a weekday, 0 otherwise
IsWeekend	Displays 1 if date is a weekend, 0 otherwise
IsHoliday	Displays 1 if date is a public religious danish holiday

Table 2.2: Attributes of the original main data set.

The BBR data set was mapped to the main dataset to get house specifications and energy labels for each `SK_Household`.

2.2 Data preparation

In this section, it will be described how the raw data was prepared for the analysis. Data preparation is essential for any statistical analysis to avoid interference from outliers or measurement errors. Since some of the data is entered by the user, the users can also cause interference. There were no observed values of `EnergySale`, so it was assumed that none of the houses had solar cells. To obtain one daily value for each observation and attribute the hourly data was either accumulated or averaged to get the proper daily values.

Variable	Type	Unit	Abbreviation	coefficient
EnergyPurchase	Continuous	kWh	EP	y
EnergyPurchaseScaled	Continuous	kWh/m^2	EPs	y_s
Squaremeters	Continuous	m^2	SM	-
AirTemperature	Continuous	°c	AT	uA
AtmosphericPressure	Continuous	Pa	AP	ap
WindSpeed	Continous	m/s	ws	ws
WindDirectionDescription	Discrete	-	W$_x$	W_x
SurfaceSolarRadiationDownwards	Continous	watt/m^2	SS	gA
Residents	Discrete	-	RE	-
BBR_YearOfConstruction	Continous	-	YOC	-

Table 2.3: Chosen variables for further analysis, where x denotes one of the eight principal winds.

Table 2.3 shows the variables kept for the analysis and in which format they are represented. The variable selection is also discussed in Chapter 3. After the data preparation, only 35 houses remained. The rest of this section will explain how the data was acquired.

The data was aggregated from hourly values into daily values. This is done to get rid of any possible dynamic within the day. `EnergyPurchase` is accumulated to the sum of the consumption during the given day. `AirTemperature`, `AtmosphericPressure`, `WindSpeed`, `WindDirectionDescription`, `SurfaceSolarRadiationDownwards` are all daily averages.

`EnergyPurchaseScaled` is the original `EnergyPuchase` divided with the number of squaremeters in the given house.

2.2.1 Conversion from degrees to directions.

A new variable `WindDirectionDescription` was created to divide the wind direction into the eight principal winds, N, NE, E, SE, S, SW, W, NW. This makes the interpretation of the variable easier as it makes categorization and comparison easier since the wind direction goes from 1° to 360°. For example, N was made from data points in the variable `WindDirectionDegree` varying from 337.5 to 22.5 degrees.

2.2.2 Date selection

Since this report will mainly focus on the classification part and not about forecasting the consumption, it was decided to remove the periods where the consumption was at a constant base level, as it is the heating patterns that are interesting. By investigating the consumption as a function of time, it is possible to see a general trend.

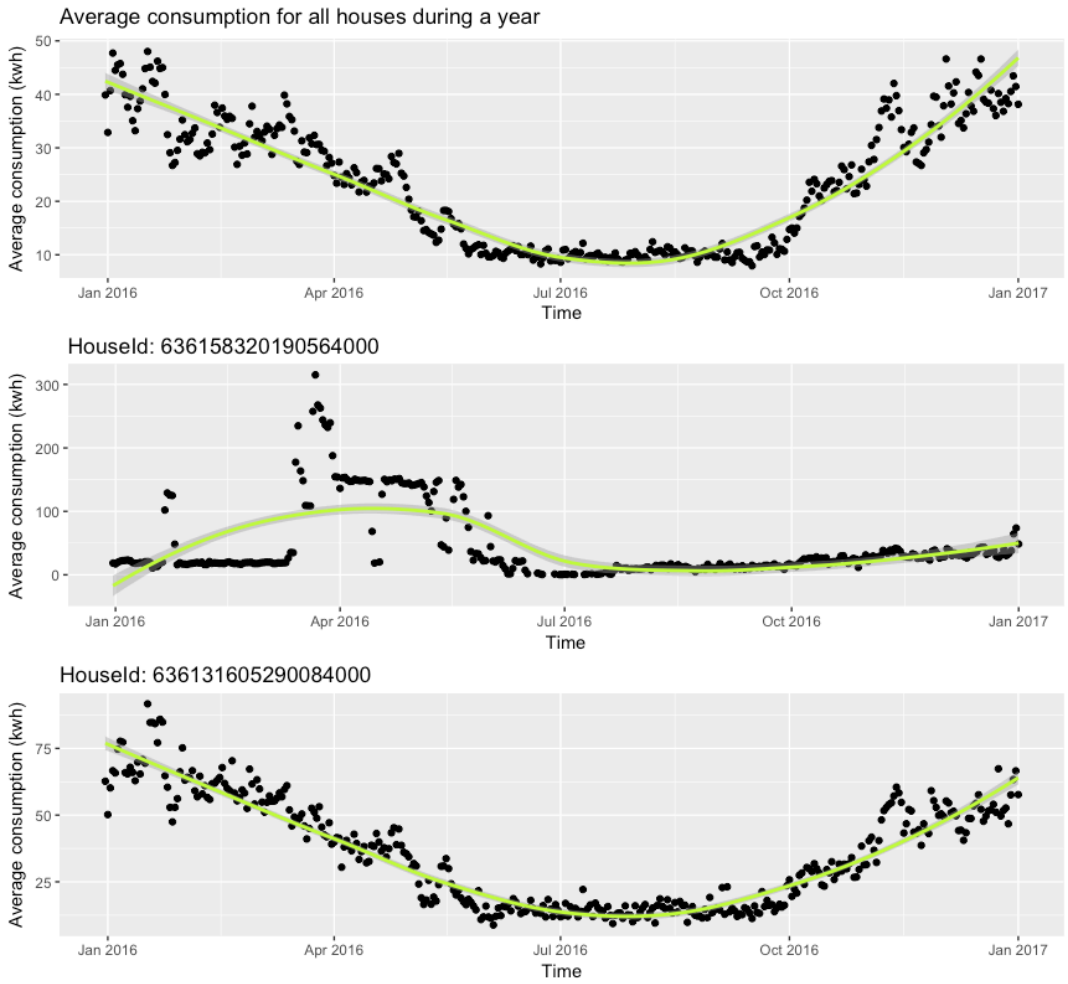

Figure 2.1: Daily consumption during 2016. The top plot shows the general trend for all houses. The middle shows an example of a house deviating from the trend. The bottom plot shows a house following the trend.

Figure 2.1 shows the daily average consumption for all houses. The slope around the warmer months is closest to 0. By removing this period, it is possible to eliminate the period with a different pattern, that may interfere with the regression analysis. To accommodate the individuality of each house, the baseline periods were found for each house by only looking at the consumption as a function of time, as in Figure 2.1. These periods were removed. Houses, where the trend deviated significantly from the general trend like the middle plot in Figure 2.1 was removed. The removed periods can be found in Table A.1 under the variables `StartTime` and `EndTime`

2.2.3 Energy labels

Today, the energy labels used in practice, range from A to G, where A has three levels. In this assignment there will be no distinguishing between the different levels of label A. Hence we continue with seven labels. [10].

Figure 2.2: Range of certified energy labels. Reproduced from [10].

2.2.4 House details

Table 2.3 shows the distribution of the house specifications from the original data. One house had 0 residents, and one house had 0 square meters. If we investigate the distribution of `EnergyLabel`, the true distribution resembles a normal distribution [13], far from the provided data. The provided energy labels do not represent the true distribution adequately. Meanwhile, the distribution of construction years looks similar to the distribution we see in Figure 2.3, [12].

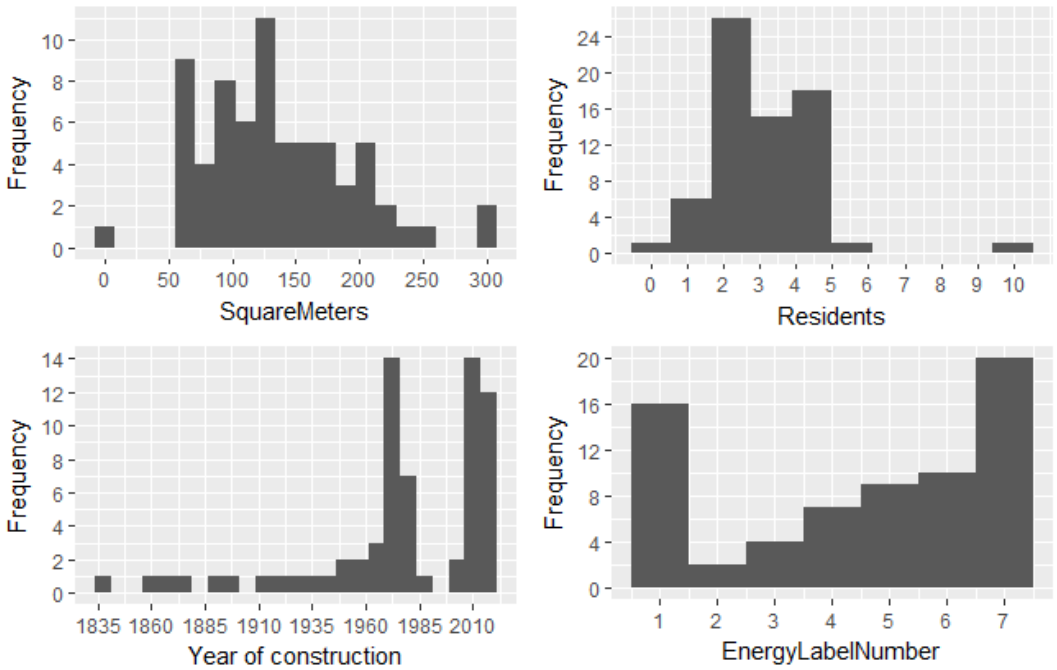

Figure 2.3: Distribution of different house details, where `EnergyLabelNumber` corresponds to `EnergyLabel` as numbers instead of letters.

2.2.5 Removed data

From the provided data, holidays were removed as they can obscure the consumption patterns. This included the removal of Pentecost, Great Prayer Day, Christ's Ascension, Easter and Christmas days and January 1st. Data with `SquareMeters`= 0, `Residents`= 0 and buildings where `BBR_HouseType` was classified as 'Holiday' or 'NonResidential' were also removed. Since only one house contained more than six residents, it was decided to combine the number of residents above five into one category denoted '6'. Buildings with unusually low consumption were also removed. This included daily purchased energy kWh/m^2 below 0.001. For example, a house of 250 m^2, with a refrigerator turned on for 24 hours daily using 100 kWh a year yields $100/365/250 \approx 0.001$ kWh/m^2 daily consumption. Of course, this number increases for smaller houses, so it seems reasonable to remove daily values below 0.001 kWh/m^2. To obtain houses with a full year of observations, houses with less than 300 daily observations were removed.

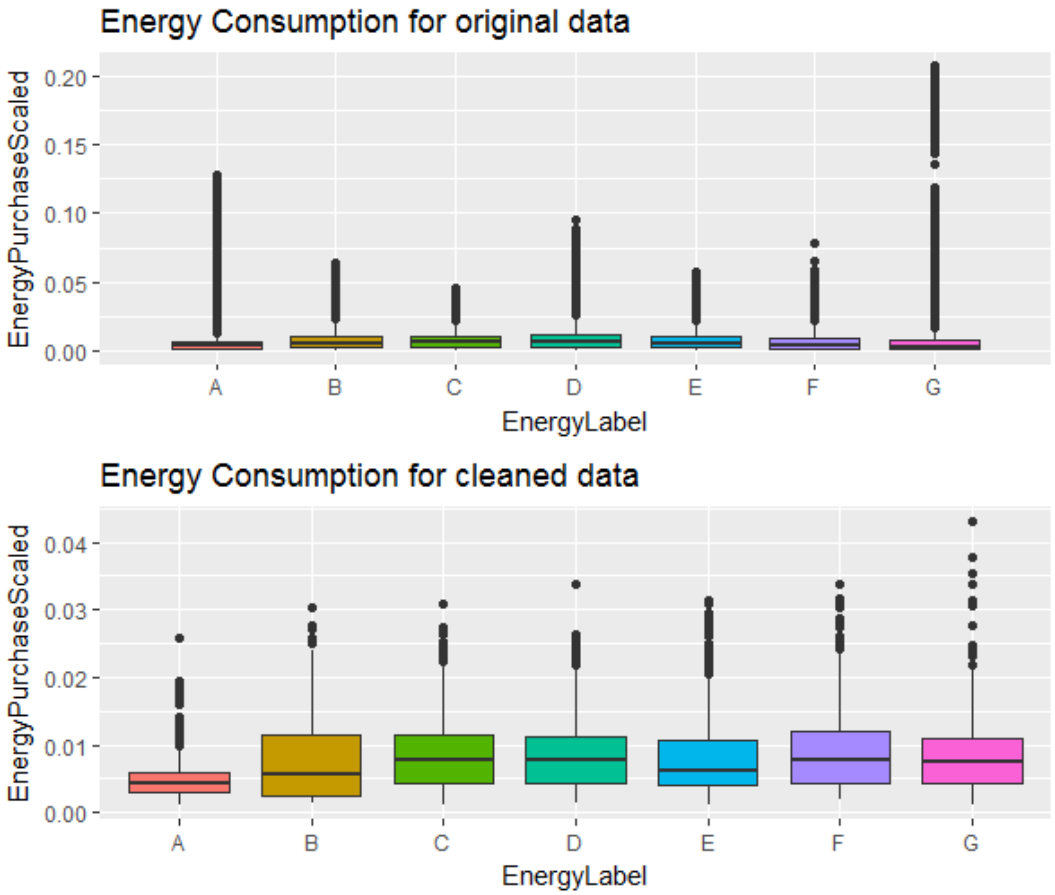

Figure 2.4: Boxplot of daily observations from 35 houses, divided into energy labels.

Figure 2.4 shows the scaled consumption for each energy label before and after the clean up described in this chapter. It is clear that the distribution improves since the skewness reduces, and we have less ”outliers.” It is expected that the consumption will increase going from label A to G. The boxplots show that the distribution of the houses lies very close to each other, except for A that, as expected, has lower values of `EnergyPurchaseScaled`. This show an indication of a non-ranked order between the labels.

Table A.1 shows house ids and BBR information of the remaining house in the regression dataset. Table A.2 shows the same for the classification data set. The main difference between the two data sets is that there is no date selection, and houses deviating from the general trend are not removed. In the next chapter, we will go further into the analysis and detect patterns in the data.

CHAPTER 3

Exploratory analysis

Exploring the patterns in the data is the first step in the analysis. This chapter is about visualizing the different variables to detect possible structures or correlations. Exploratory analysis is also used to specify the data modeling due to pattern discovery. It is highly necessary for the regression analysis, as several assumptions need to be fulfilled.

The temperature is considered, from basic physical knowledge, the most significant variable in the level of heating in a house. From this, it is expected that there exists a significant relationship between the energy consumption and the outside temperature.

3.1 Examination of `AirTemperature`

The air temperature is inspected. Figure 3.1 shows `EnergyPurchaseScaled` as a function of `AirTemperature`, divided into each of the 7 energy labels. A simple linear regression model is fitted to each, to see the steepness of the slope (simple linear regression will be explained in Section 4.1). It is expected that the steepness of the slopes will increase as we go from energy label A to G because it is expected that the lower energy labels will use more energy as the temperature decreases compared to the higher energy labels. Figure 4.1 does not seem to show this pattern as surprisingly, B has the steepest slope, but A does have the lowest. The slope coefficients for C, D, E, F are very similar. Precaution should be taken, as energy label, B only contains two houses, Figure 2.3. Another thing that can be seen is that there does not seem to be a difference in the consumption based on weekdays or weekends. The weekends seem to be as evenly scattered out as weekdays.

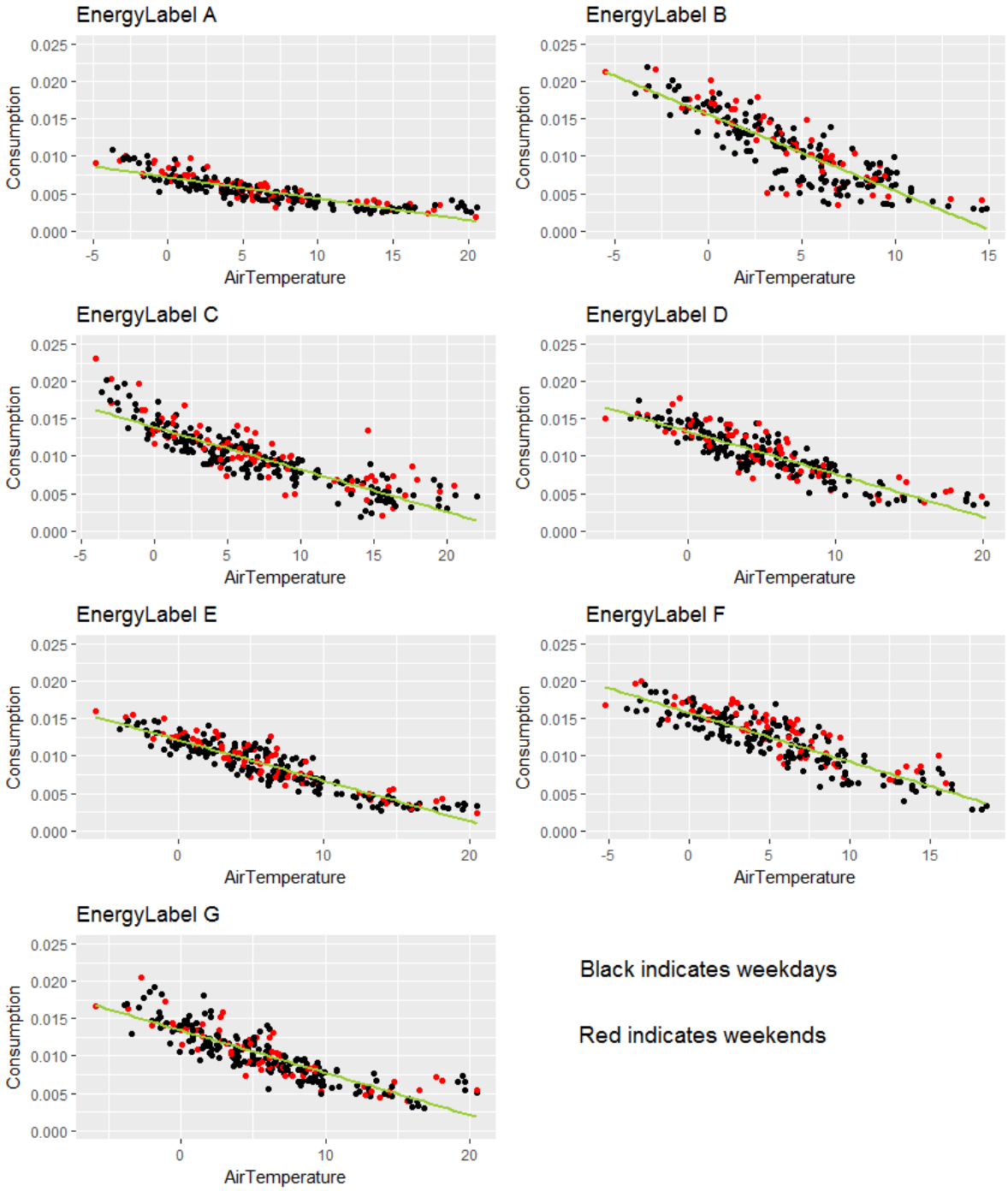

Figure 3.1: `EnergyPurchaseScaled` as a function of `AirTemperature` divided into the 7 energy labels. The green line shows the slope of a linear model. The slope coefficients are A = -0.000287, B = -0.00103, C = -0.000568, D = -0.000572, E = -0.000546, F = -0.000655, G = -0.000566.

3.2 Weather data

The weather data is individually examined before the data modeling begins. In section 4.1 it is discussed how a linear relationship between the dependent and independent variable is desired. This can be investigated through a scatterplot between the two. If a non-linear relationship is detected, one should try a transformation on either the dependent or independent variable to obtain linearity.

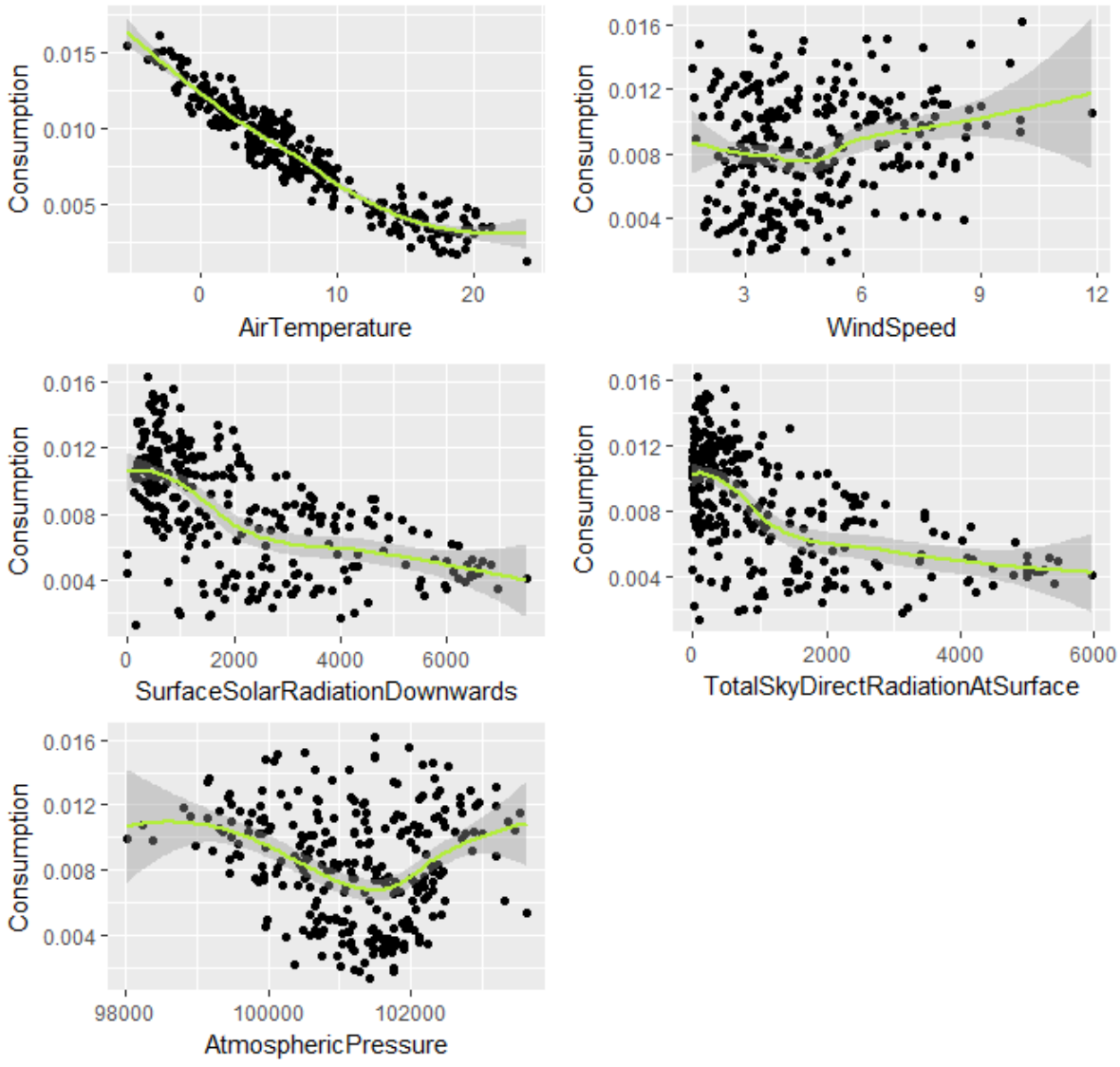

Figure 3.2: Scatterplots of the daily mean values of the meteorological variables. The green line is a smoothing line based on the conditional means.

Figure 3.2 shows a scatterplot between `EnergyPurchaseScaled` and each meteorological variable. Since Figure 3.2 shows the daily mean of all houses, the patterns for each individual house can be completely different, but investigating Figure 3.2 shows that only `AtmosphericPressure` or maybe `AirTemperature` could use a polynomial transformation. Since `AtmosphericPressure` is so randomly scattered, it is decided not to use a transformation. As for the `AirTemperature`, the slope seems to even out as it increases, but these points have been removed from most houses, so it is expected that the relationship is linear between these variables. Thus, no transformation is performed for any of the houses.

3.3 Multicollinearity

Multicollinearity is caused by having explanatory variables being highly linear dependent with each other. When working with linear regression models and classification methods, a simpler model, is always preferred, but there are other issues related to multicollinearity. In linear regression, multicollinearity can cause a term to be insignificant when it is highly related to the dependent variable. When two explanatory describes the same, the interpretation of the estimates can become unstable and the standard error increases. Multicollinearity can be checked by calculating the correlation between each independent variable using the function `cor()`. The problem can be solved by removing one of the correlated variables without loss of information. The correlation ranges from -1 to 1, where ± 1 corresponds to two identical variables. Table 3.1 shows that the correlation between

	AT	AP	ws	TSDRAS	SS
AT	1.00	0.04	-0.11	0.51	0.56
AP	0.04	1.00	-0.25	0.06	0.01
ws	-0.11	-0.25	1.00	-0.23	-0.25
TSDRAS	0.51	0.06	-0.23	1.00	0.96
SS	0.56	0.01	-0.25	0.96	1.00

Table 3.1: Correlation between each continous explanatory variable. TSDRAS is an abbreviation of `TotalSkyDirectRadiationAtSurface`.

`TotalSkyDirectRadiationAtSurface` and `SurfaceSolarRadiationDownwards`, is very close to 1. It is decided to remove the variable `TotalSkyDirectRadiationAtSurface`, while the others from Table 3.1 is kept, both in the regression dataset as well as the classification data set.

3.4 Interactions

An interaction describes the combined effect between two or more independent variables on the dependent variable. Interactions can expand models, as they allow the combination of explanatory variables to contribute to the response variable. For example, it could be that when the weather is cold and windy, it will result in a higher consumption compared to the marginal effect of `AirTemperature` and `WindSpeed`. In this section, different possible interactions will be examined to see if they have a combined effect, that may contain significant information about `EnergyPurchaseScaled`. Only two-way interactions will be considered, meaning the effect from two independent variables.

It seems reasonable to expect that the interaction between `WindSpeed` and `AirTemperature` is significant because the effect of a cold wind is more significant than the wind from warm weather. Figure 3.3 shows that the energy consumption as a function of windspeed and temperature does not have a combined effect, because as `WindSpeed` increases, `EnergyPurchaseScaled` only changes as a function of `AirTemperature`. The same pattern can also be detected for the interaction between `AirTemperature` and `SurfaceSolarRadiationDownwards`. If the interaction between `WindSpeed` and `WindDirection` is considered, random scatter seems to be the general trend, hence no evidence of interaction. Figure 3.3 only gives an indication of possible interactions, but it can not confirm nor reject the possibility. Since Figure 3.3 is solely based on the daily average of all houses, it does not mean that the individual houses follow the same pattern. As it is wanted to obtain one model that fits all houses, individual interactions will not be considered.

In the further analysis, the interaction between `AirTemperature` and `WindSpeed` will be considered since it is physically reasonable.

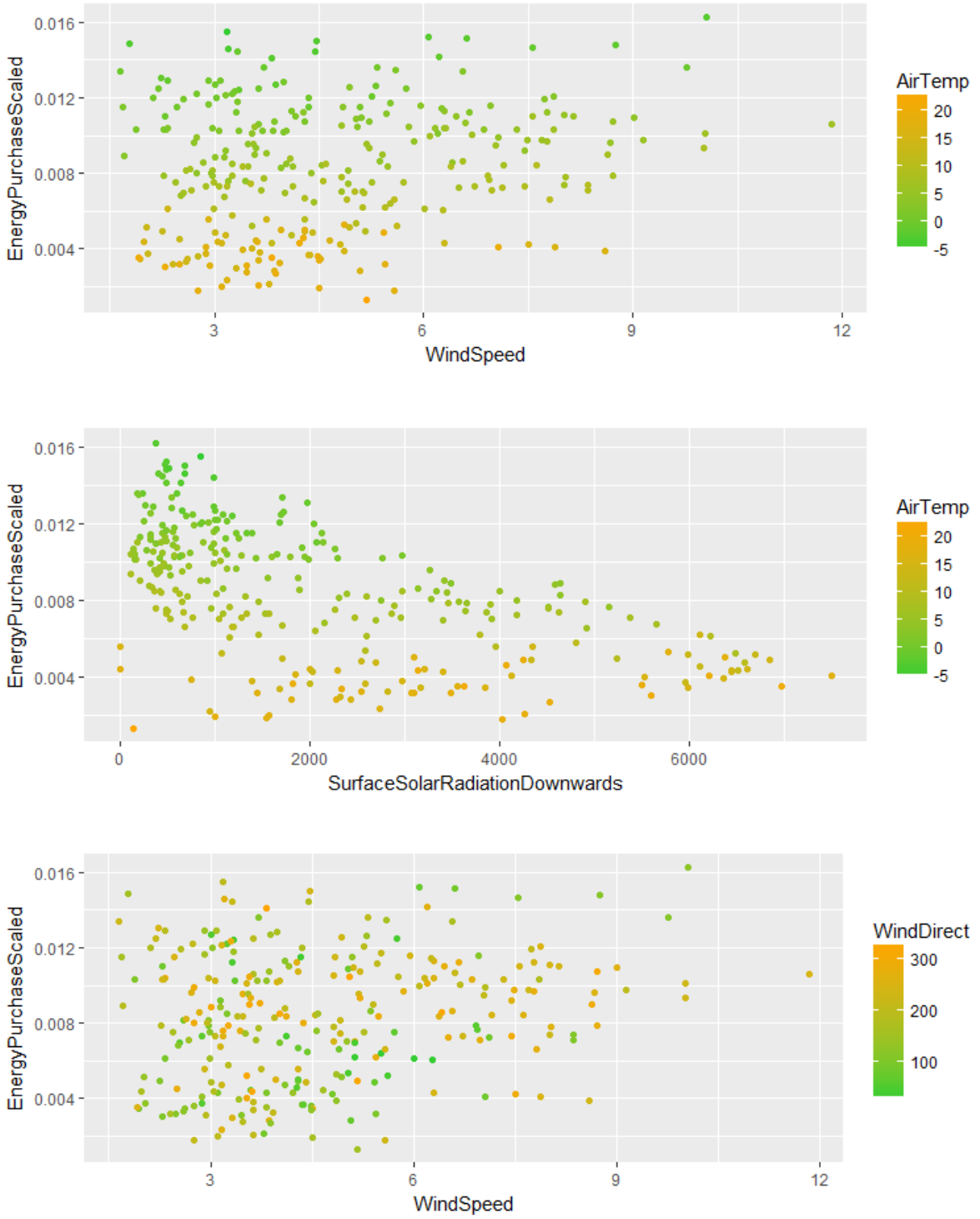

Figure 3.3: Scatterplots with color mapping of the average daily observations off all houses including the variables `SurfaceSolarRadiationDownwards`, `AirTemperature`, `WindSpeed` and `WindDirectionDegree`.

CHAPTER 4

Regression

4.1 Linear regression

Linear regression is a method that describes the relationship between a dependent continuous variable Y and one or more independent variables $x_1, x_2, ..., x_n$. Linear regression can be useful for predicting outcomes of Y given different observations of x_j, but in this report, it will be used to determine the relationship and strengths of the various estimates of β_j. By determining the contribution of each independent variable β_j, a linear regression model can be formulated as

$$Y_i = \beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i} + ... + \beta_p x_{p,i} + \varepsilon_i \qquad \varepsilon_i \sim \mathcal{N}(0, \sigma^2) \qquad i \in (1, 2, ..., n) \quad (4.1)$$

Where ε_i are the errors assumed to be i.i.d(independent and identically distributed). We distinguish between simple and multiple linear regression, where simple only has one explanatory variable, and multiple regression has two or more. The model in Equation 4.1 shows a multiple regression model with p independent variables.

To get the best linear model Y, it is necessary to obtain the smallest total deviation from the actual data. This can be done several ways, but it is chosen to use least squares method given as

$$\text{RSS} = \sum_{i=1}^{n} (Y_i - (\beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i} + ... + \beta_p x_{p,i}))^2 \quad (4.2)$$

$$= \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2 \quad (4.3)$$

Hence it is the parameters β_j that is optimized to minimize RSS. When this is done, the underlying assumptions that need to be fulfilled must be checked to validate the model. The assumptions can be summarized as

- Linearity between Y and x_j
- Independent residuals
- Homogenic variance amongst residuals
- Normally distributed residuals with mean 0 and variance σ^2

If the assumptions are not met, it can influence the predictions. We refer to Chapter 2 and 3 in [26] for more theory on the subject and methods to validate the models. It is not always the case that all the parameters contribute to describing the relationship between Y and x. When doing statistics, simpler models with fewer parameters that describe the structure and patterns are always preferred. There are many ways to select the most significant variables and different criteria to determine the order of included and excluded variables.

Variable selection is an iterative process that updates the model for each step of the variable selection based on a specific criteria. Chapter 7 in [26] explains the concept and approach of forward/backward selection. To summarize, the forward selection is a method that starts with the simplest model and adds the terms that are most significant, while the backward selection is the opposite. It starts with the complete model including all terms and drops the variables that are least significant. In this report, forward selection will be used. There are advantages and disadvantages of both methods. Since the methods can only go back and forth, the final model may include/exclude terms that turn out to be insignificant/significant, because changing a model also changes the level of significance of all variables. The level of significance can be determined by different methods; here the Akaike information criterion score is applied, AIC, Chapter 7 in [26], or an F-test which only selects variables with a given p-value below some threshold, Chapter 6 in [15]. The AIC score is based on the maximum likelihood, while the F-test is based on the variance between two models.

In this report, both a simple and multiple linear regression is performed. By interpreting the estimates of the relationship between `EnergyPurchase` and different explanatory variables it is possible to see which meteorological phenomenon affects the efficiency of a specific house. We will be using R-squared. R-squared is a measure of fit, specifically the variability of the response variable in the given model (Chapter 5 [26]). Usually, a high R-squared (ranges from 0 to 1), is desired. However, several other things need to be considered, so it will only be used as an indicator.

4.2 The simple linear regression model

A linear regression model will be fitted to each of the remaining houses to determine what affects the energy consumption. First the simple linear model, including `EnergyPurchaseScaled` as a function of `AirTemperature`, will be investigated. Next, all the variables mentioned in Table 2.3, excluding the BBR data, will be used to create multiple linear regression models using the function `lm()` from [25]. Different transformations of the response variable will be considered, as discussed in Chapter 3. The models will then be validated based on the assumptions stated in Chapter 4.1.

Since the simple linear model only includes one explanatory variable, there will be no variable selection. The models are constructed for each house, and the estimates are found. Different transformations of the response are tried, and a log transformation turns about to be the transformation that increases the mean R-squared the most. In the multiple regression, the square root transformation turned out to be the transformation which increased the R-squared the most. To be able to compare the simple and multiple regression the square-root transformation is chosen for both models even though the log transformation was better in the simple linear regression analysis.

The model to be tested for each house is then:

$$\sqrt{Y_s} = \mu + \beta_{uA} \cdot x_{\text{AT}} + \varepsilon \tag{4.4}$$

4.2.1 Validation

Different validation methods including a QQ-plot, Residuals vs. Fitted plot, sign-test and Kolmogorov-Smirnov test will be used. The QQ-plot and Residuals vs. Fitted plot are, as explained in [26], plots, where it can be seen whether or not the residuals are i.i.d with $\mathcal{N}(0, \sigma^2)$.

To summarize, in a QQ-plot all the residuals should lie on a straight line to fulfill normality, and the Residuals vs. Fitted plot should show that the residuals are randomly scattered around mean 0 to satisfy constant variance. A sign test is a non-parametric test that counts the number of sign changes, here amongst the residuals. It is assumed that $\mu = 0$, so the sign test expects that over 47.5% of the residuals will have a positive sign and over 47.5% will have a negative sign (assuming a 95% confidence interval). The Kolmogorov-Smirnov test, `ks.test()` from the package [25], is another non-parametric test that can compare two sample distributions. For more information on the two tests and how to calculate the level of significance see Chapter 3 in [15].

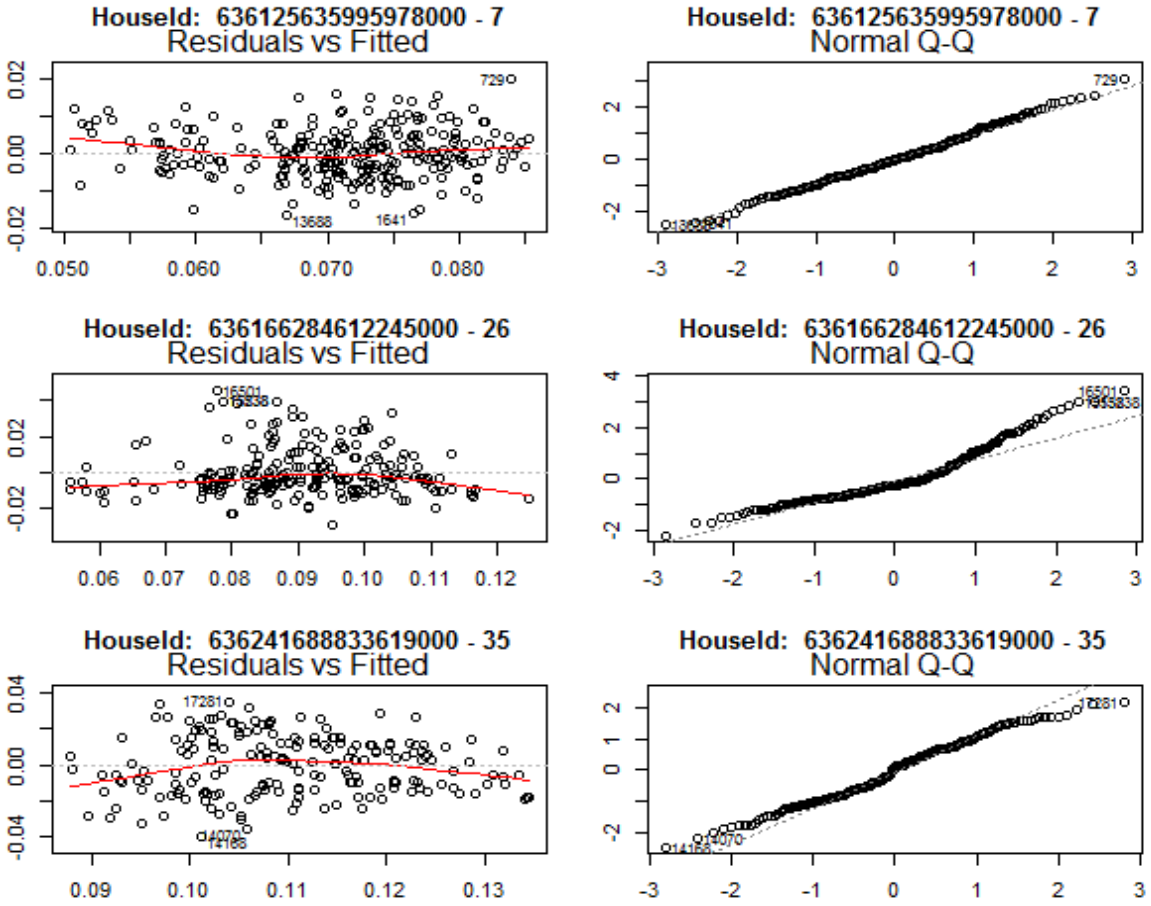

Figure 4.1: Residual plots of selected houses based on the simple linear model in equation 4.4. The assumptions of a linear model are violated in the two bottom figures, while the top figure is acceptable.

Figure 4.1 shows examples of some of the houses that do not fulfill the assumptions and an example of how it should look like. The top plot shows that the residuals are randomly scattered around the mean 0, and a nice straight QQ-plot. It is expected that the tails are always a bit skewed. The middle and bottom plots show that the residuals are not normally distributed. The middle QQ-plot indicates that the distribution is skewed, while the bottom middle plot shows a non-straight QQ-line. The residuals in House 26 seem to show a bit of a pattern as many of the residuals above 0 have a higher spread, while it should be more randomly scattered. Thus the variance is not homogeneous. Overall most of the residuals plots did seem to fulfill the assumptions. It is possible to correct some of the models that are not valid, by using other transformations, but we keep the models as they are to stay in the same domain.

Table A.4 shows the results from a sign- and Kolmogorov-Smirnov test. It is clear that the sign test barely passes any of the houses. This does not mean that the residuals do not resemble white noise since the magnitude of each residual is not taken into account, but like all the other tests and plots it gives an indication of the residuals and if they

meet the underlying assumptions. If we look at the ks-test, we see the opposite. Most of the models pass the test.

To summarize our observations, most of the simple linear models seem sufficient. However, there is room for improvements in some of them. It is clear that much information can not solely be described by `AirTemperature` as it does not capture all of the variability in the data.

4.2.2 Results

Figure 4.2 shows the estimates of uA as well as a 95% confidence interval. What should be noted, is that the confidence intervals are quite large given the domain that is being examined. Many of the estimates have wide confidence intervals crossing 0. This means that in practice the outside temperate effect can not be estimated accurately due to noise (variation).

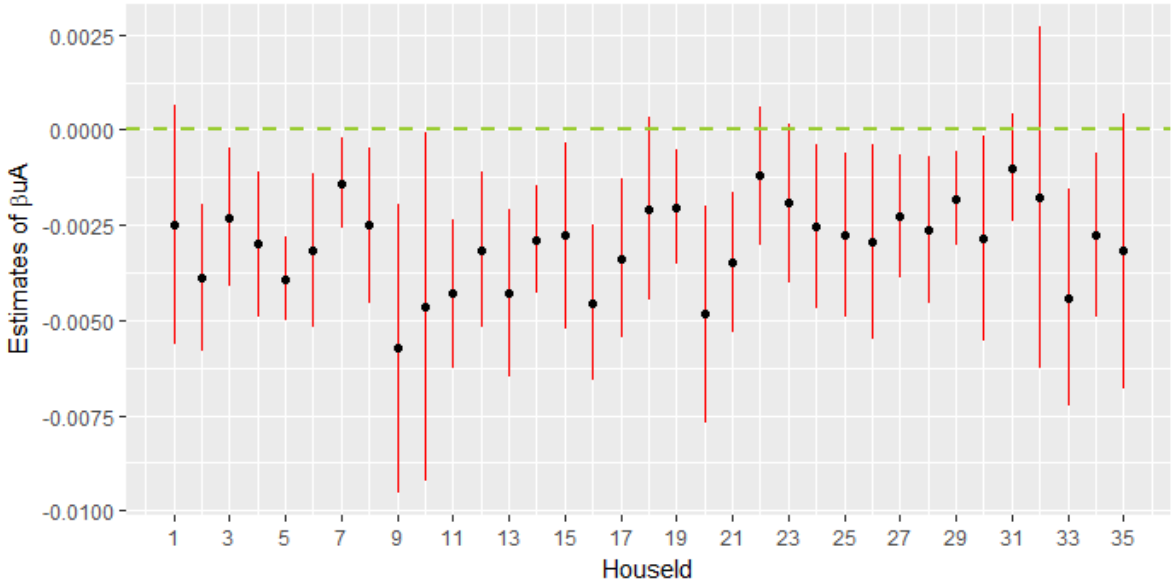

Figure 4.2: Estimates of β_{uA} coefficients for the simple linear regression model. The model returns a mean R-squared = 0.556.

Table A.3 shows the estimates from all models. The squared intercept can be interpreted as the baseline consumption when `AirTemperature` is 0. When working with a square root transformation of the response, it is challenging to interpret the magnitude of the

estimates directly as

$$\begin{aligned}\sqrt{\hat{Y}_s} &= \mu + \beta_{uA} \cdot x_{\text{AT}} &\Leftrightarrow\\ \hat{Y}_s &= (\mu + \beta_{uA} \cdot x_{\text{AT}})^2 &\Leftrightarrow\\ \hat{Y}_s &= \mu^2 + (\beta_{uA} x_{\text{AT}})^2 + 2\mu\beta_{uA} x_{\text{AT}}\end{aligned}$$

Including more variables quickly increases the complexity. As mentioned, the sign of the estimates is beneficial to explain, and since 35 houses are involved, the magnitude can be compared to the other houses.

4.3 The multiple linear regression model

The simple linear regression is now extended, so it is possible to include or exclude some of the patterns found in Chapter 3. The purpose is to improve the current simple models and include more variables to make the distinguishing between houses clearer. We discussed the idea of only carrying out transformations on the response and including an interaction between `AirTemperature` and `WindSpeed` in Chapter 3. Based on the residual plots in Figure 4.1, different variance stabilizing transformations can be considered, including a logarithmic and square root transformation, a squared transformation was also examined. As earlier mentioned, we only consider forward selection with either an F-test or AIC as criteria. To summarize the process, it can be described in pseudo-code as

Algorithm 1 Implementation of alternative model selection

```
Input: data set, list of variables, list of houses, list of days to remove
Output: list of significant estimates and standard deviation for each house
Choose if EnergyPurchase should be scaled or not
Choose model selection criteria AIC/F-test
Choose transformation of response
    for each house
        Compute simple linear regression model using lm()
            for each variable set
                Calculate AIC/F-test using add1()
                Choose variable with lowest AIC/p-value
                 if AIC no longer decreases/p-values are above 5% then
                    break
                 Update model with new variable
                 repeat for remaining variables in the set
     return current model estimates and standard deviation
end
```

This means that there are models that do not include all variables. It is not possible to include `Residents` or `BBR_YearOfConstruction`, since these values are the same within

each observation in each house, hence will not have an impact in a linear model.

It was discovered that the combination of choosing a square root transformation and F-test as criteria yielded the highest R-squared. It also solved the variance inhomogeneity and non-normality issues. In the following sections, the analysis of the multiple linear regressions model is continued with the chosen transformation. As will be discussed, the interpretation is difficult when doing transformations, hence, in Section 4.5 the estimates of a multiple linear regression with no transformation will be presented to get a proper interpretation of the results.

4.3.1 Validation

The multiple linear regression models will again be validated the same way as the simple linear regression models. Figure 4.3 show the same 3 houses as in Figure 4.1.

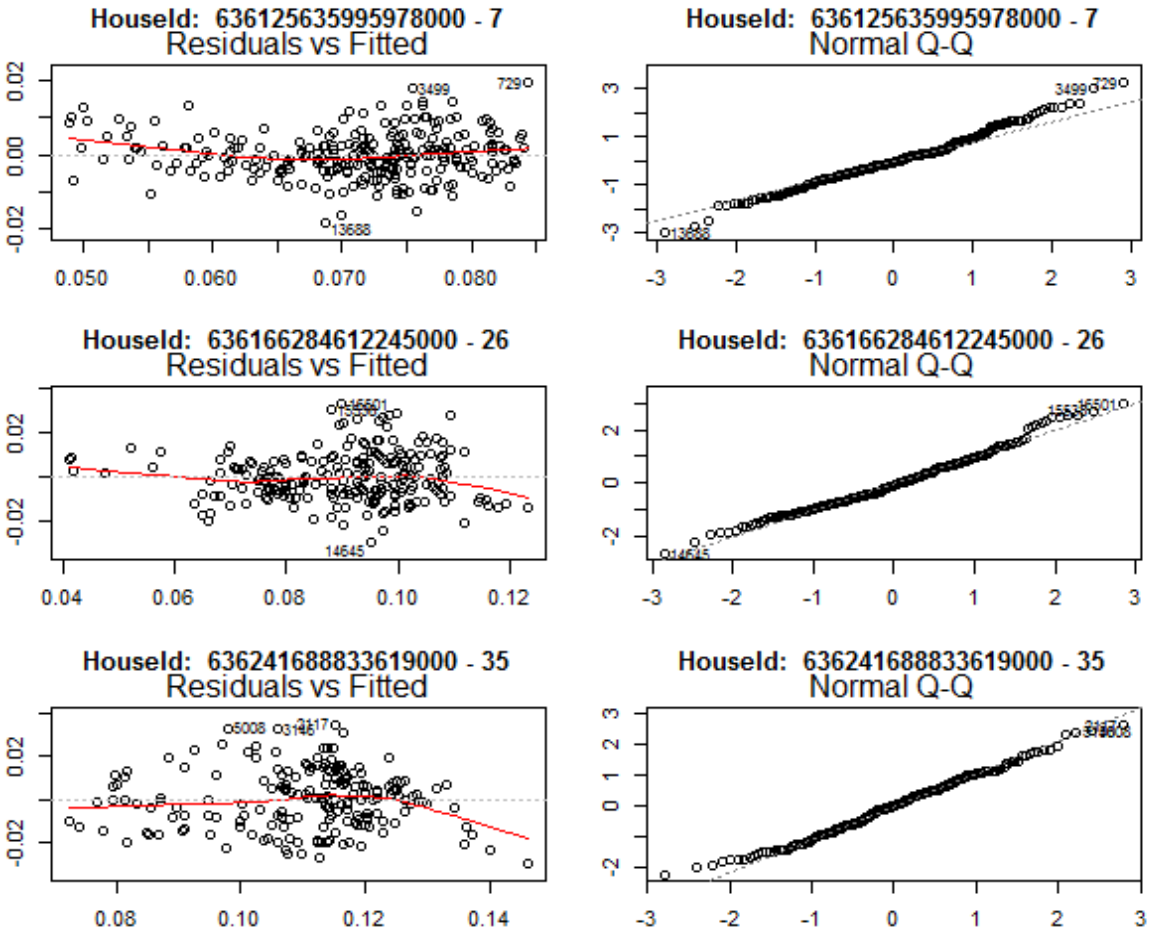

Figure 4.3: Residual plots of selected houses based on the multiple linear regression models with a square root transformation.

It can be seen that the normality of HouseId 26 and 35 have improved since the QQ-plots are less curved. However, there is still room for improvement in the middle house. Meanwhile, the variance seems to be constant. The general trend for all of the houses shows that almost all of the houses have improved. Going from a simple to a multiple linear regression model does not always necessarily improve the residuals.
Table A.4, containing the sign and ks-test, shows there is not much change in any of the two tests, but if the residual plots are taken into account as well, it is safe to say that in most of the cases we do meet the assumptions.

4.3.2 Results

The final estimates of the multiple linear regression models are now ready to be presented. Table 4.1 shows the results of the analysis while Figure 4.4 shows some of the key results in this Chapter. There is quite a lot of information that can be drawn, but first, we will have a look at the model structure to be evaluated. As mentioned the highest R-squared is obtained when a square root transformation is applied

$$\begin{aligned}\sqrt{\hat{Y}_s} = \mu &+ \beta_{uA} \cdot x_{\mathrm{AT}} + \beta_{gA} \cdot x_{\mathrm{SS}} + \beta_{ws} \cdot x_{\mathrm{ws}} + \beta_{ap} \cdot x_{\mathrm{AT}} + \beta_{ws:AT} \cdot (x_{\mathrm{AT}} \cdot x_{\mathrm{ws}}) \\ &+ \beta_{W_{NE}} \cdot I_{W_{\mathrm{NE}}} + \beta_{W_{NW}} \cdot I_{W_{\mathrm{NW}}} + \beta_{W_S} \cdot I_{W_{\mathrm{S}}} + \beta_{W_{SE}} \cdot I_{W_{\mathrm{SE}}} \\ &+ \beta_{W_W} \cdot I_{W_{\mathrm{W}}} + \beta_{W_N} \cdot I_{W_{\mathrm{N}}}\end{aligned} \tag{4.5}$$

where I_x is an indicator function $I_x \rightarrow \{1, 0\}$. The intercept, μ, can be interpreted as the energy consumption when all other variables are set to 0. In practice, this scenario will never occur, so one should not take the intercepts too literally. For example, it will never occur that the atmospheric pressure is 0.

In general, we see that, as expected, the estimates of β_{uA} are all negative, meaning, when the temperature outside is high, the consumption decreases. Also, when the wind speed increases, the consumption increases since almost all of the estimates are positive. What seems to be a bit odd, is that two coefficients of β_{gA} are positive and one β_{ws} coefficient is negative. This means that two houses increase its' consumption when the weather is sunny, and one house uses less electricity when it is windy. The estimates of β_{gA} and β_{ap} might seem very small, but the domain in which they are measured is quite high, so when adding this term to the linear model, the contribution can be quite significant. Very few of the houses were not affected by the wind speed while even fewer were not affected by the sun. As was discussed in Section 3.4, the interaction between `Windspeed` and `AirTemperature` did not seem significant. Table 4.1 also shows that very few houses found this interaction significant.

Like simple linear regression, when interpreting the estimates, one should always be cautious because the marginal effects do not always tell the full story and the square root transformation should also be taken into consideration. However, the signs of the estimates can explain most of the interpretation, but the magnitude is difficult to

interpret. Comparing the estimates to the other houses is a possible way to say something about the magnitude.

Id	μ	β_{uA}	β_{gA}	$\beta_{W_{NE}}$	$\beta_{W_{NV}}$	β_{W_S}	$\beta_{W_{SE}}$	$\beta_{W_{SV}}$	β_{W_V}	β_{ws}	β_{W_N}	β_{ap}	$\beta_{AT:ws}$
1	0.111	-0.002	-2.16e-06	0.007	0.004	0.005	0.001	0.013	0.008				
2	0.106	-0.004	-2.19e-06	0.001	-0.006	0.002	0.001	0.001	-0.002	0.001			
3	0.103	-0.002	-3.13e-06										
4	0.095	-0.003		0.001	0.007	0.008	0.004	0.009	0.010	0.002			
5	0.032	-0.004		-0.002	-0.005	-0.000	0.002	-0.000	-0.000	0.001	-0.007	8.57e-07	
6	0.122	-0.003	-3.18e-06	-0.006	-0.003	0.003	-0.001	0.003	-0.000	0.001	-0.007		
7	0.188	-0.001	-1.40e-06							-0.000		-1.03e-06	
8	0.108	-0.002	-3.45e-06							0.001			
9	0.147	-0.005	-5.92e-06										
10	0.508	-0.004	-5.41e-06									-3.69e-06	
11	0.251	-0.004	-7.43e-07	-0.003	0.006	0.009	0.005	0.010	0.007	0.002		-1.21e-06	
12	0.095	-0.003	-2.86e-06										
13	0.137	-0.004	-2.30e-06	-0.013	-0.007	-0.004	-0.002	-0.008	-0.007		0.001		
14	0.209	-0.002	-2.30e-06									-9.25e-07	
15	0.074	-0.001	-1.22e-06							0.004			-4.10e-04
16	0.148	-0.004	-2.61e-06							0.002			
17	0.106	-0.003	-1.78e-06							0.001			
18	0.086	-0.001	-2.68e-06							0.001			
19	0.078	-0.002	-3.04e-06										
20	0.140	-0.005	-5.49e-06	0.004	-0.002	0.006	0.005	0.006	0.005	0.002	-0.010		
21	0.102	-0.003	-3.16e-06	-0.003	-0.003	0.001	0.002	0.004	0.001		-0.003		
22	0.090	-0.001		-0.001	0.002	0.005	0.005	0.005	0.008				
23	0.089	-0.002	-4.09e-06	0.001	0.005	0.007	0.004	0.005	0.008		0.015		
24	0.115	-0.002	-2.37e-06										
25	0.113	-0.002	-1.33e-06							0.002			-1.37e-04
26	0.104	-0.002	-4.13e-06	-0.004	0.007	0.007	0.002	0.003	0.008				
27	0.091	-0.002	-1.62e-06										
28	0.109	-0.003								0.002			
29	0.079	-0.002	-1.32e-06										
30	-0.106	-0.003	-4.86e-06									2.04e-06	
31	0.070	-0.001	1.12e-06							0.001			
32	0.118	-0.002	-3.42e-06										
33	0.103	-0.002	-2.35e-06							0.006			-5.52e-04
34	0.107	-0.003	-1.35e-06	-0.006	0.006	0.005	0.004	0.004	0.006	0.001	0.002		
35	0.109	-0.006	1.58e-06	0.001	0.006	0.009	0.007	0.012	0.015	0.001	-0.005		3.74e-04

Table 4.1: Estimates from the multiple linear regression model. Including a square root transformation of the response, and an F-test as model selection criteria. All estimates in the table has a p-value below 0.05%. W_E is the reference point. Some of the houses including W as a variable does not include W_N as no wind was observed from this direction.

To show an example of how to use Equation 4.5 we will consider an observation from HouseId 1 where EPs = 0.0098, AT = 7.04, AP = 101566.43, ws = 3.43, W_x = S, SS= 3311.83.

Using the estimates for this house, from Table 4.1 yields the model:

$$\begin{aligned}
\hat{Y_s} &= (\mu + \beta_{uA} \cdot x_{\text{AT}} + \beta_{gA} \cdot x_{\text{SS}} + \beta_{ws} \cdot x_{\text{ws}} + \beta_{ap} \cdot x_{\text{AT}} + \beta_{AT:ws} \cdot (x_{\text{AT}} \cdot x_{\text{ws}}) \\
&\quad + \beta_{W_{NE}} \cdot I_{W_{\text{NE}}} + \beta_{W_{NW}} \cdot I_{W_{\text{NW}}} + \beta_{W_S} \cdot I_{W_{\text{S}}} + \beta_{W_{SE}} \cdot I_{W_{SE}} \\
&\quad + \beta_{W_W} \cdot I_{W_{\text{W}}} + \beta_{W_N} \cdot I_{W_N})^2 \\
\hat{Y_s} &= (0.111 - 0.002 \cdot x_{\text{AT}} - (-2e-06) \cdot x_{\text{SS}} + 1 \cdot \beta_{W_S} \\
&= (0.111 - 0.002 \cdot 7.04 - (-2e-06) \cdot 3311.83 + 0.005)^2 \\
&= 0.108^2 \\
&= 0.011
\end{aligned}$$

close to the observed value of 0.0098.

It can be seen that East is not included in the formula, as this can be considered as the reference direction, so if there is an observation from East, no additional term will be added to the model.

Figure 4.4 shows the estimates of the most significant variables, in which, the estimates can be compared with each other. House 35 seems to be a bit of an outlier as the β_{uA} value is quite low, with a significant variance, the β_{gA} estimate is positive, and the confidence interval for β_{ws} crosses 0. However, the residuals in Figure 4.1 did now show any sign of violations of the assumptions.

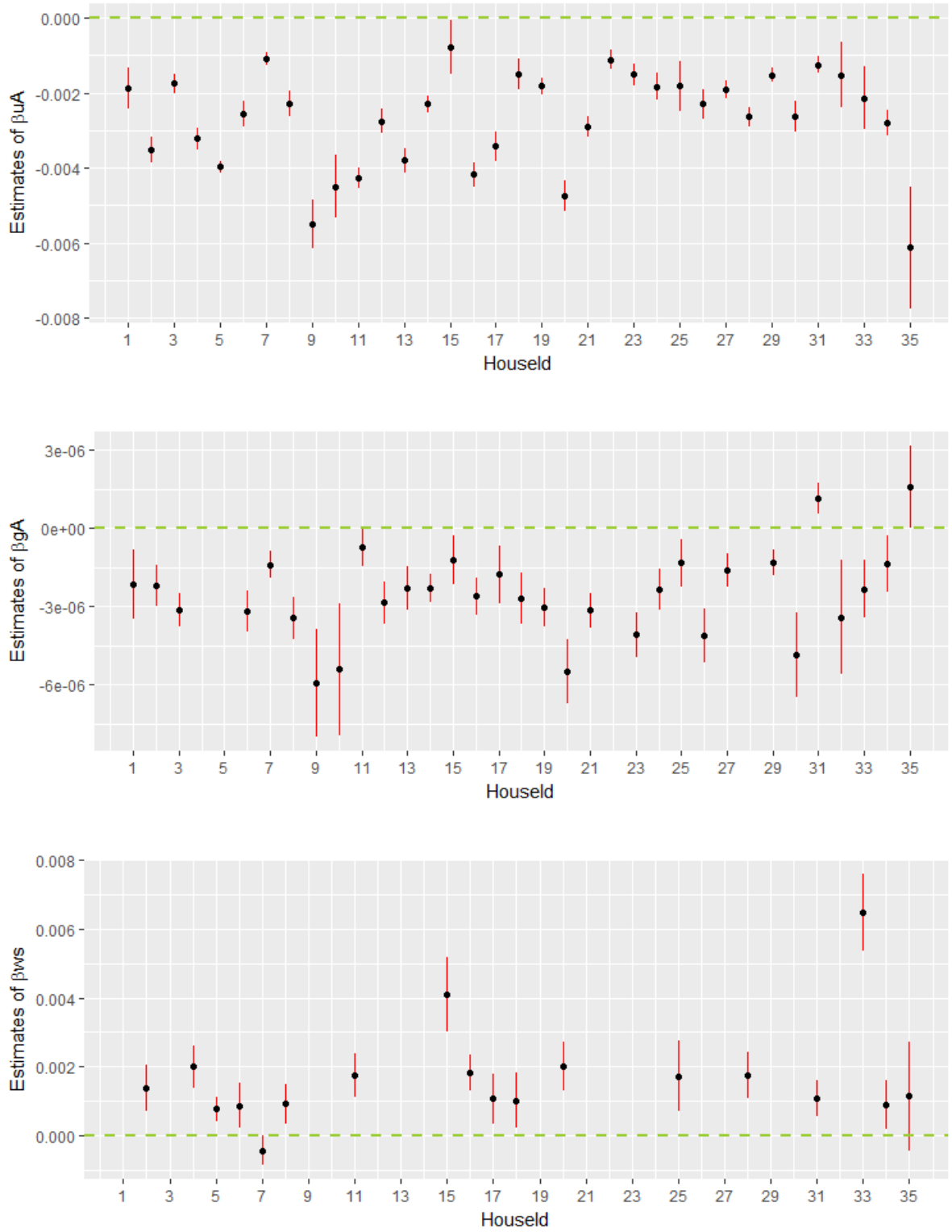

Figure 4.4: Estimates of β_{uA}, β_{gA} and β_{ws} coefficients, including 95% confidence interval. The parameters are found using a square root transformation of the response. The F-value was used as model selection criteria. The mean R^2 was found to be 0.651. Estimates that were insignificant are left out.

4.4 Comparison

In our case, it can be seen that the β_{uA} estimates are lower for the multiple linear regression compared to the simple as we have other variables which describe the consumption. Also, the variance between the estimates of β_{uA} is much bigger for the simple linear regression. Figure 4.2 shows that the estimates are in the range of each other while Figure 4.4 shows are much bigger spread, and also less uncertainty. It is clear that the houses that had poor performance in the simple linear regression were also the some of the worst in the multiple linear regression. The mean R-squared increased from 0.556 to 0.651. None of the houses decreased their R-squared going from simple to multiple regression.

We were able to perform a box-cox transformation on all models, which did improve the mean R-squared to be 0.763. However, we need the estimates to come from the same domain to continue with the values for clustering, but also to be able to compare them individually. The box-cox method has its benefits in it can find the best transformation that optimizes the maximum likelihood, in this case of a transformation of the response, to obtain normality [26]. Finding one overall model structure that fits all houses is the biggest problem with the regression analysis.

Overall, the simple linear regression can give a quick overview of the performance of each house. However much information about the house performance can not solely be described by the β_{uA} coefficient.

4.5 Results of the non-transformed multiple linear regression

To get a better interpretation of the estimates and performance of the houses, Table 4.2 shows the estimates in the original domain. Figure A.1 shows that excluding the transformation yields violations of the linear regression assumptions, as the variance of the residuals show a funnel shape.

As expected, all intercepts and uA values are positive. One house shows a positive gA value and one house shows a negative ws value. In both analyses of the multiple linear regression models, house 31 showed a positive gA value.

Id	μ	uA	gA	ws	Id	μ	uA	gA	ws
1	0.0138	-3.57e-04	-4.30e-07		18	0.0074	-2.46e-04	-3.70e-07	1.70e-04
2	0.0114	-6.73e-04	-3.62e-07	2.41e-04	19	0.0061	-2.38e-04	-3.92e-07	
3	0.0105	-2.96e-04	-5.19e-07		20	0.0210	-1.19e-03	-1.42e-06	4.90e-04
4	0.0103	-5.94e-04		4.17e-04	21	0.0105	-5.05e-04	-4.93e-07	
5	0.0142	-7.96e-04		1.16e-04	22	0.0092	-1.83e-04	-1.40e-07	
6	0.0149	-5.82e-04	-5.95e-07	2.12e-04	23	0.0088	-2.41e-04	-6.28e-07	
7	0.0065	-1.63e-04	-1.56e-07		24	0.0131	-3.68e-04	-3.97e-07	
8	0.0117	-4.58e-04	-6.29e-07	1.89e-04	25	0.0137	-4.59e-04	-2.65e-07	1.79e-04
9	0.0211	-1.29e-03	-1.17e-06		26	0.0116	-3.91e-04	-7.41e-07	
10	0.0165	-1.11e-03	-9.15e-07	3.61e-04	27	0.0083	-2.99e-04	-2.22e-07	
11	0.0172	-1.02e-03		5.67e-04	28	0.0122	-5.19e-04		3.32e-04
12	0.0089	-4.47e-04	-3.59e-07		29	0.0062	-2.08e-04	-1.57e-07	
13	0.0172	-8.49e-04	-3.25e-07		30	0.0110	-4.03e-04	-8.27e-07	-2.03e-04
14	0.0132	-4.50e-04	-3.74e-07		31	0.0050	-1.69e-04	1.42e-07	1.51e-04
15	0.0067	-4.31e-04		3.70e-04	32	0.0117	-4.72e-04		3.83e-04
16	0.0216	-1.12e-03	-5.21e-07	5.08e-04	33	0.0130	-1.06e-03	-4.68e-07	9.84e-04
17	0.0114	-6.43e-04	-3.48e-07	1.86e-04	34	0.0119	-5.32e-04	-2.58e-07	2.14e-04
					35	0.0126	-7.90e-04		6.06e-04

Table 4.2: Estimates from the multiple linear regression model. Excluding transformations and using an F-test as model selection criteria. All estimates in the table has a p-value below 0.05%. The mean R^2 was found to be 0.607. All units are as in Table 2.3.

Using the same example as is Section 4.3.2 we can use the estimates from Table 4.2 to calculate the scaled energy consumption

$$\hat{Y}_{\text{s}} = 0.0138 - 0.000357 \cdot x_{\text{uA}} - 0.000000430 \cdot x_{\text{gA}} \tag{4.6}$$
$$= 0.0138 - 0.000357 \cdot 7.04 - 0.000000430 \cdot 3311.83 \tag{4.7}$$
$$= 0.01093363 \tag{4.8}$$

Since no transformation is performed, the estimates can be interpreted directly. For example, for every degree increase in the air temperature, the consumption will increase with 0.000357. If the temperature is below 0, the consumption per degree will decrease with 0.000357.

4.6 Multiclass logistic regression

Linear regression models can be extended into Generalized linear models, where the properties and assumptions are different. Generalized linear models can be used when the residuals are not $\mathcal{N}(0, \sigma^2)$. Logistic regression is a generalized linear model that can formulate a model with a discrete outcome. This becomes handy when we want to predict `EnergyLabel` for each house. Two different generalized linear models will be used:

- Ordered logistic regression, similar to multinomial logistic regression, but assumes that the classes are ordered.
- multinomial logistic regression, a classification method with multiple classes as the outcome.

The energy labels are expected to be ordered, meaning we assume that there is a natural order between the classes, i.e., `EnergyLabel` A is expected to do better than `EnergyLabel` B, `EnergyLabel` B is expected to do better than `EnergyLabel` C and so on. Figure 3.1 showed `EnergyPurchase` as a function of `AirTemperature` for each class of `EnergyLabel`. There was no evidence of a clear ordering of performance for the different energy labels because the steepness of the regression lines did not decrease as we went from class A to G. Hence both methods will be considered.
Because of the ordering, the ordered logistic regression can be formulated as a cumulative probability

$$P(Y \leq j) = \frac{\exp(\alpha_j - \beta x)}{1 + \exp(\alpha_j - \beta x)} \qquad j \in (1, 2, ..., J) \tag{4.9}$$

$$\text{logit}(P(Y \leq j)) = \alpha_j - \beta x \tag{4.10}$$

This is know as the proportional odds model of ordered logistic regression. Each marginal probability in the ordered logistic regression can also be computed by

$$P(Y = j) = P(Y \leq j) - P(Y \leq j - 1) \tag{4.11}$$

Chapter 6 in [8] explains the theory of multinomial and ordered logistic regression. It also explains the concept of a *link function*. The link function allows a model with non-linear response to be connected with the linear terms in a linear regression model. One could rewrite Equation 4.1 into a generalized linear model such that we obtain

$$f(\mu) = \beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i} + ... + \beta_p x_{p,i} + \varepsilon_i \tag{4.12}$$

where f is the link function. In Equation 4.14 and 4.9 the logit function is used as the link function. The choice of using logit is used to interpret of the estimates (log-odds). The two logistic regression method uses maximum likelihood to find the estimates of β, see [8] for calculation of the likelihood.

Because of the model structure in Equation 4.9, the assumptions of the proportional odds model includes parallel slopes, as β is constant for any class j. This also means that the log odds of $Y \leq j$ are linearly related to each X (Chapter 13 [18]). The log odds is given as

$$\log(\text{odds}) = \log\left(\frac{P_j}{1 - P_j}\right) \tag{4.13}$$

where $P_j = P(Y \leq j)$.

The multinomial logistic regression can be formulated as

$$\begin{aligned} P(Y=j) &= \frac{\exp(\alpha_j - \beta_j x)}{1+\sum_h^{J-1}\exp(\alpha_h - \beta_h x)} \qquad j \in (1,2,...,J) \\ \log\left(\frac{P(Y=j)}{P(Y=J)}\right) &= \alpha_j - \beta_j x \end{aligned} \tag{4.14}$$

Where $P(Y=j)$ is the probability of class j, and J is the baseline model (reference group). Here, the logit function is also used as the link function. Notice that the denominator is the same for any class j, hence $\sum_j P(Y=j) = 1$, Chapter 6 [8]. Instead of working with odds, the relative risk ratio is considered which is given as

$$\text{RR} = \frac{P(Y=j)}{P(Y=i)} \qquad j \neq i \tag{4.15}$$

The most significant difference between the multinomial and ordered logistic regression is that the estimates of β in the ordered logistic regression are the same for any class j, while it changes in the multinomial model. In Chapter 5 more classification methods will be considered.

4.6.1 The logistic regression models

For the logistic regression analysis, the data described in Section 5.1 is used instead of the regression data, because a comparison with the other classification methods is wanted. Hence, the variables are normalized. The reason why will be explained later. The ordered logistic model will be fitted using the function `polr()` and the multinomial will be fitted using `multinom()` from [29]. To keep the classification algorithms consistent, there will not be performed any variable selection or transformation. This will not be necessary as logistic regression does not require any linear relationship between the dependent and independent variables

To verify the model, we can plot the cumulative probabilities by fixing the continuous variables at their means, fixing the discrete variables at some factor and letting one variable vary, or by verifying that the log odds can be written as a linear combination for any choice of x, in this case, any choice of `Residents`. The log odds can be calculated by Equation 4.9 for all levels of `Residents`, then calculating Equation 4.11 followed by the log odds from Equation 4.13. This results in the logs odds found in Table A.7. The table shows that the assumption of a linear combination is almost met with a few remarks.

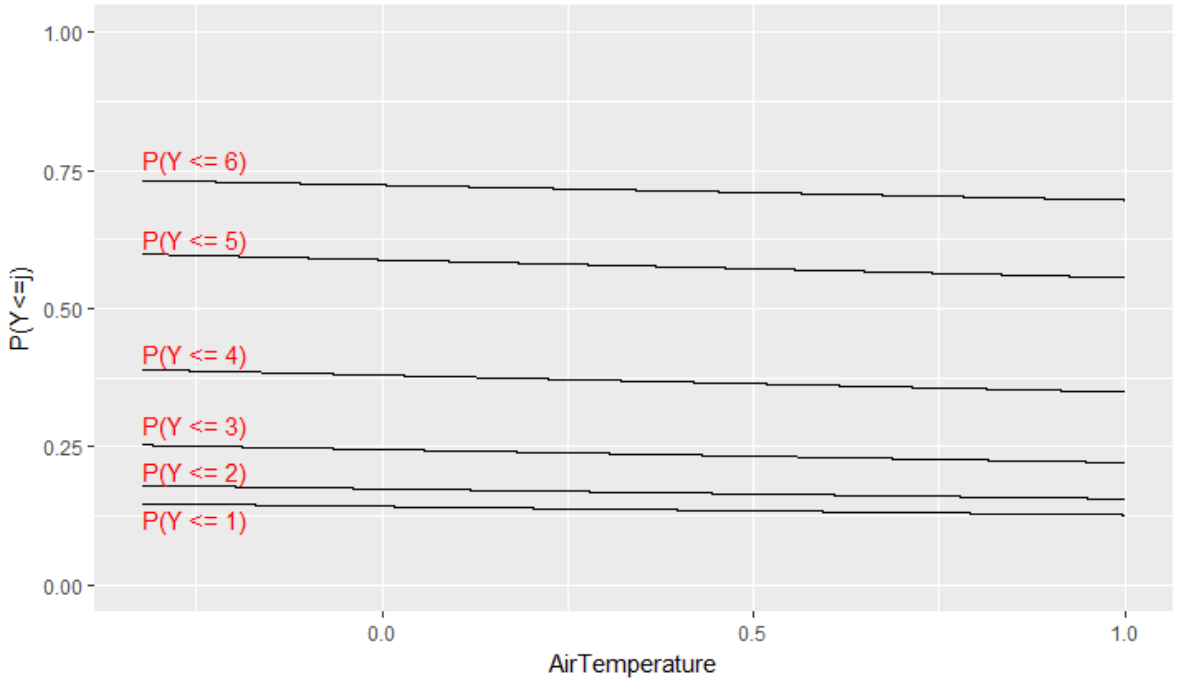

Figure 4.5: The proportional odds model as a function of `AirTemperature`. All variables are fixed at their mean and `Residents = 2`.

Figure 4.5 shows the cumulative probability of the classes as a function of `AirTemperature`. The slopes are parallel. However, it can be seen that the temperature does not seem to change the probabilities at all.

Based on Figure 4.5, it seems like, it can be assumed that the assumption of ordinal classes is fulfilled. Another way to test this is through a likelihood ratio test, Chapter 6 in [24], between the ordered and multinomial model. The LRT compares the likelihood of two models, where one is nested within the other. LRT is given as

$$\text{LRT}: \qquad G = -2\log\left(\frac{\text{likelihood(model1)}}{\text{likelihood(model2)}}\right) \tag{4.16}$$

where model1, in this case, is the nested ordered logistic regression model, and model2 is the multinomial regression model. Following a χ^2 distribution with $p_1 - p_2 = 72 - 17 = 55$ degrees of freedom yields a p-value close to 0. Hence, the hypothesis of reducing the multinomial model into the less complex ordered logistic regression model is rejected.

Table A.5 and A.6 shows the estimates of the models. Like linear regression, there is a reference point to the multinomial estimates. Each column corresponds to a model equation $\alpha + \beta X$ and all of them are compared to `EnergyLabel = A`. The ordered regression model has the same estimates for all coefficients, but different intercepts.

To show an example of the ordered logistic regression, the mean of all normalized variables with `Residents = 2`, `EnergyLabel = B` and using the estimates from Table A.6 yields the ordered logistic regression model and cumulative probability:

$$
\begin{aligned}
P(Y \leq j) &= \frac{\exp(\alpha_j - \beta x)}{1 + \exp(\alpha_j - \beta x)} \\
z_j &= \alpha_j + \beta x \\
z_B &= -7.39 - (0.132 \cdot x_{\text{AT}} + 0.05 \cdot x_{\text{EPs}} - 0.239 \cdot x_{\text{ws}} - 0.356 \cdot x_{\text{SS}} \\
&\quad + \beta_{\text{RE=2}} - 5.973 \cdot x_{\text{AP}}) \\
&= -7.39 - (0.132 \cdot (-0.339) + 0.05 \cdot 0.06 - 0.239 \cdot 0.331 - 0.356 \cdot 0.299 \\
&\quad + 0.414 - 5.973 \cdot 0.974) \\
&= -1.848 \\
P(Y \leq B) &= \frac{\exp(z_B)}{1 + \exp(z_B)} \\
&= \frac{\exp(-1.759)}{1 + \exp(-1.759)} \\
&= 0.147
\end{aligned}
$$

Hence, the cumulative probability of being in class B or below is 14.7 %. To find the probability for any other class, only the intercept, α_j changes. The marginal probabilities can simply be calculated by using Equation 4.11.

To interpret the estimates, it can be much easier to consider the odds ratio by taking the exponential of the estimates, [8]. By assuming every variable is fixed, a 1-unit increase can be explained by the odds.

Variable	OR	2.5 %	97.5 %	p-value
AirTemperature	1.1411	0.9798	1.3289	0.0896
EnergyPurchaseScaled	1.0508	0.5317	2.0767	0.8866
Residents2	1.5121	1.3769	1.6607	0.0000
Residents3	0.8316	0.7569	0.9138	0.0001
Residents4	0.4712	0.4299	0.5165	0.0000
Residents5	1.3335	1.1462	1.5514	0.0002
Residents6	1.3232	1.1127	1.5735	0.0015
Residents10	0.0000	0.0000	0.0000	0.0000
WindSpeed	0.7872	0.6304	0.9829	0.0347
SurfaceSolarRadiationDownwards	0.7002	0.6116	0.8015	0.0000
AtmosphericPressure	0.0025	0.0001	0.0604	0.0002

Table 4.3: Odds Ratio of the estimates in the ordered logistic regression. Where EnergyLabel = A and `Residents = 1` is the reference group.

For example, if we consider a 1-unit increase in `EnergyPurchaseScaled`, holding the other variables fixed, the odds of being in a higher class than A (going towards G) is increased with 5%. The same can be said for `AirTemperature`, if we increase 1-unit, the odds of being a higher class than A is 14% when assuming the assumptions are held. If we look at the odds of going from `Residents = 1` to `Residents = 10`, the odds of being a higher class than A is almost 0. This is probably because there was only one house with ten residents, which happened to be of class A. Increasing `WindSpeed` decreases the odds of a higher class by 22%. It is a bit surprising that neither `AirTemperature` and `EnergyPurchaseScaled` is not significant.

To compare the number of Residents in each house we can look at the odds ratio between different groups. Using the cumulative probabilities (Equation 4.13), the odds ratio can be calculated for any group j by

$$\text{odds}_j = \frac{P_j}{1 - P_j} \tag{4.17}$$

$$\text{OR}_{ji} = \frac{P_j}{1 - P_j} \Big/ \frac{P_i}{1 - P_i} \qquad j \neq i \tag{4.18}$$

Taking `Residents = 1` and `Residents = 3` yields the odds ratio

	A|B	B|C	C|D	D|E	E|F	F|G
OR:	1.141070859	1.050831785	0.831640499	0.787187846	0.700161878	0.002546166

Table 4.4: Odds ratio between `Residents = 1` and `Residents = 3`

For `Residents = 1`, the estimated odds of being class A or B rather than a higher class is 1.14 times the odds compared to `Residents = 3`. Or taking another example, the estimated odds of being class D or below rather than a higher class is 0.83 times the odds for `Residents = 1` compared to `Residents = 3`
The same pattern can be seen for the other combinations of residents. Succinctly, houses with many residents tend to be in a higher class of energy label than houses with fewer residents.

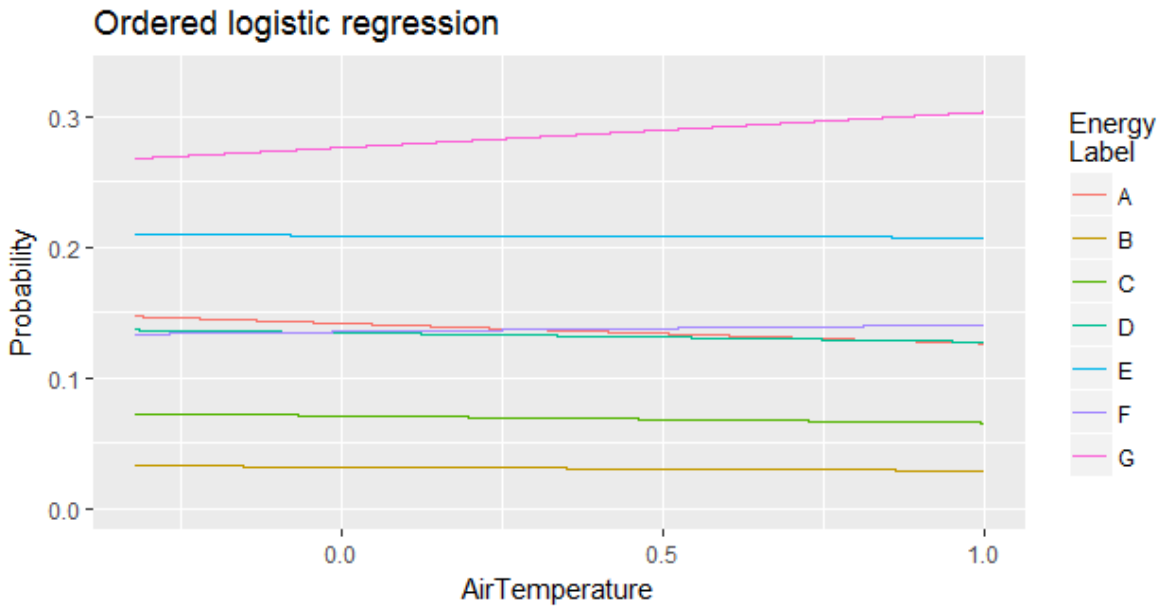

Figure 4.6: Ordered logistic regression, where all the variables are set to the mean and `Residents = 2`. The variables have been normalized.

Table 4.6 Shows the probability of being in each class as a function of `AirTemperature`. It can be seen that `AirTemperature` does not have any impact on the probabilities.

For clarification, the multinomial model is also shown. Figure 4.7 shows the probabilities as a function of `AirTemperature` and Table A.5 shows the estimates and P-values. Each column corresponds to a model that compares two levels of the dependent variable, here the reference class is A. The estimates can be interpreted almost the same way as the ordered logistic model, by taking the exponential of the estimates, but instead of having odds we consider risk ratios. Table 4.5 shows the relative risk of the multinomial logistic regression model. It can be seen that for a 1-unit increase in `AirTemperature` (all other variables are fixed) the relative risk of being label B compared to label A is 1.36 times higher.

Energy Label	B	C	D	E	F	G
(Intercept)	3.361e+02	1.0372+06	5.417e+12	3.979e+14	4.141e+08	1.151e+05
AirTemperature	1.362e+00	1.177e+00	2.487e+00	1.652e+00	2.195e+00	1.054e+00
EnergyPurchaseScaled	1.159e+03	5.338e+02	3.885e+03	1.718e+02	4.408e+03	3.931e-01
Residents2	3.125-06	1.352e-01	5.823e-08	1.630e-07	1.100e-13	1.923e+01
Residents3	1.041e+00	3.555e-02	1.390e-14	1.286e-07	3.932e-08	5.264e+00
Residents4	8.615e-01	2.737e-02	2.329e-08	6.307e-08	6.466e-08	1.094e+00
Residents5	2.857e-07	4.504e-09	1.225e-14	8.752e-17	1.619e-07	4.121e+00
Residents6	3.168e+00	1.311e+00	2.609e-07	1.916e+00	5.671e-06	2.256e+01
Residents10	1.209e-06	3.498e-08	1.694e-13	3.624e-15	6.759e-14	2.126e-07
WindSpeed	7.436e-01	6.919e-01	4.384e-01	4.035e-01	1.217e+00	5.111e-01
SS	1.974e+00	1.924e+00	7.614e-01	9.080e-01	1.779e+00	4.465e-01
AtmosphericPressure	3.514e-04	3.670e-06	1.689e-06	9.952e-09	5.479e-03	1.564e-06

Table 4.5: Relative risk of the multinomial logistic regression model, where label A is the reference class.

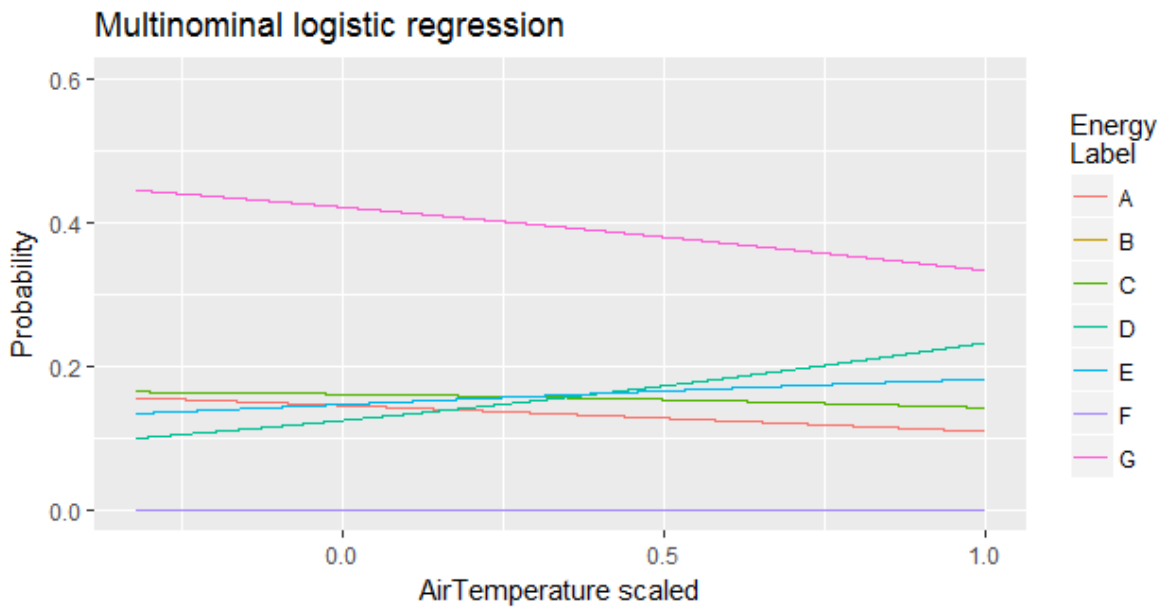

Figure 4.7: Nominal logistic regression, where all the variables are set to the mean and `Residents = 2`. The variables have been normalized.

The logistic regression model has now been validated. If we look at the two plots, of the logistic regression models it is possible to see that in the ordered logistic regression model the probability of classifying a house as label B is minimal, and Figure 4.7 shows that the possibility of acquiring label F is 0. It is presumably because none of the observations did have the combination of 2 residents and label F or G. Also, both models are more likely to classify a house as label G, which happens to be the most common label in the given data set, Table 2.3. In Section 5.2.4 the performance of the logistic regression models as a classifier will be investigated.

CHAPTER 5

Classification

Up until now we have examined the significant meteorological variables and on how these affect the consumption. The energy label of the house has been briefly introduced in Section 4.6.1. In this chapter prediction of the energy label of the houses will be examined with the use of three new methods.

5.1 Methodology

Machine learning is a branch of modern statistics. Where regular statistical analysis focus on descriptive statistics machine learning focus on the predictive analysis. What is important for the machine learning methods is to be as accurate as possible and to predict as correctly as possible. Machine learning is divided into two branches, unsupervised and supervised learning. Unsupervised learning algorithms deal with an unknown response variable. Clustering methods are used to explore structures in the data, that is not already labeled the "true" class. Supervised learning algorithms deal with known response variable. In this, it is possible to calculate different performance measurements for each algorithm.

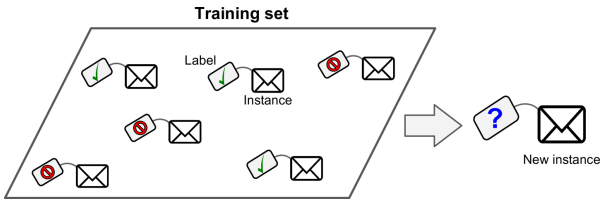

Figure 5.1: Supervised learning with two classes. A new instance can be predicted according to the known outcome of the others. Reproduced from [14]

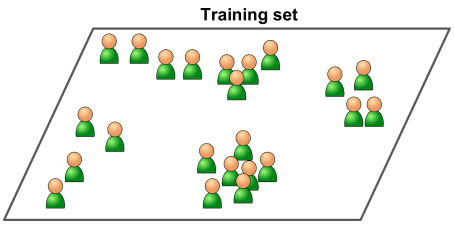

Figure 5.2: Unsupervised learning. The outcome is unknown so clustering algorithms is applied to group observations. Reproduced from [14]

In the following section, it has been chosen to use the accumulated data no matter the different patterns the houses had. A summary of the data can be seen in Table A.2. This decision is based on the interview *"Big data is better data"* held by Kenneth Cukier in which he explains how more data lets us see things that we were not able to see before. [4]. Other than working with the hourly observations, houses with residents and square meters equal to 0 were removed. Also, houses with a daily consumption below 0.001 were removed, for the same reason as mentioned in Section 2.2.5. The data has been normalized with the largest value of the given variable. Only houses with

electric heating are considered. The variable `BBR_YearOfConstruction` was originally included, but after running a few tests, an accuracy of 100 % was obtained. When including `BBR_YearOfConstruction`, each observation was too specific due to the lack of variation in data. Hence, the classification data set included the following attributes: `AirTemperature`, `AtmosphericPressure`, `WindSpeed`, `EnergyPurchaseScaled`, `Residents` and `SurfaceSolarRadiationDownwards`.

When applying the classification methods to the data, it can be necessary to use some tools that train and validates the algorithms. In the next section, cross-validation and confusion matrices are explained.

5.1.0.1 Cross-validation

Cross-validation is a machine learning technique where the data set is split into two subsets containing a training set and a test set. Our data set has been split into 90 % training set and 10 % test set. Cross-validation is a technique used to choose the optimal model relative to the error and model complexity. The cross-validation technique is only used in the k-nearest-neighbor and the artificial neural network.

For each time the algorithm runs it computes a training error in percentage meaning the difference between the predicted and the actual values. As the complexity increase for the model to be more accurate, the training error will decrease. To chose the optimal model the test error must be at its lowest. The error rate as a function of the model complexity can be seen in Figure 5.3.

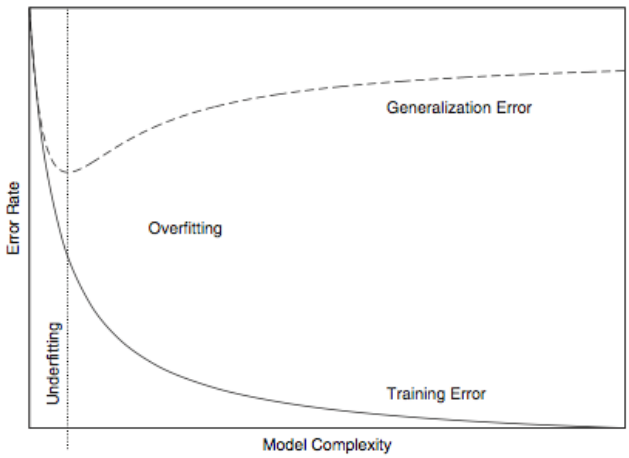

Figure 5.3: Error rate as a function of the model complexity. Reproduced from [23].

The model will underfit if a model complexity below the lowest generalization error and will overfit if a model complexity above the lowest generalization error is chosen. The terms are visualized in Figure 5.3. The model with the lowest generalization error will be selected to further analysis.

5.1.0.2 Confusion matrix

A confusion matrix is a tool, which is used for evaluating the performance of an algorithm. The confusion matrix is used when the response variable is categorical. It is usually used when the response variable is binary but can be extended to a multiclass response variable. To get a grasp of how a confusion matrix can be visualized see Figure 5.4 and Figure 5.5.

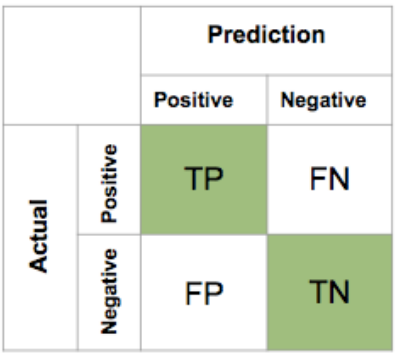

Figure 5.4: Binary

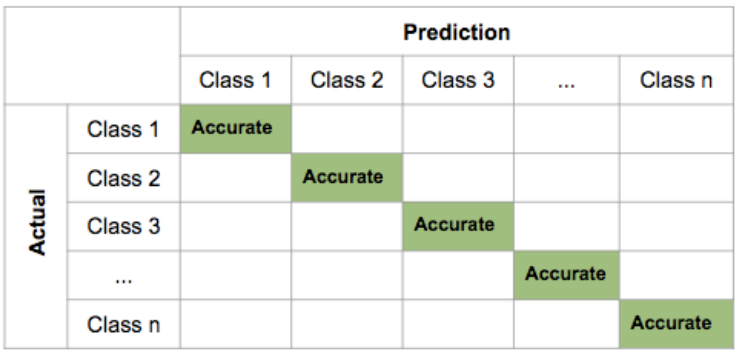

Figure 5.5: Multiclass

Figure 5.6: Example of a binary and multiclass confusion matrix. Reproduced from [3].

Diagonal elements represent correctly predicted values from the classifier. Off-diagonal elements represent incorrect predicted values. The overall performance estimate of the classifier is determined by accuracy.
The accuracy is defined as:

$$A = \frac{T_P + T_N}{T_P + T_N + F_P + F_N} \qquad A = \frac{\sum_i M_{ii}}{\sum_{ij} M_{ij}} \tag{5.1}$$

Precision is the fraction of how many of the data points which was predicted as i that was i.

$$P = \frac{T_P}{T_P + F_P} \qquad P_i = \frac{M_{ii}}{\sum_j M_{ij}} \tag{5.2}$$

Where M_{ii} represents the diagonal elements and M_{ij} represents element in the given row j.

The recall is the fraction of how many of the data points which was actually in class i which was also predicted as class i. Recall is calculated the following way

$$R = \frac{T_P}{T_P + F_N} \qquad R_i = \frac{M_{ii}}{\sum_i M_{ij}} \tag{5.3}$$

where M_{ii} represents the diagonal elements and M_{ij} represents the elements in the given column j.

To summarize; precision answers the question: "Out of the items that the classifier predicted to be in class i, how many are truly in class i?" and recall answers the following question: "Out of all the items that are truly in class i, how many are found by the classifier?" [30]

5.2 Methods

In the following methods, three different machine learning methods will be examined: Decision tree (DT), k-nearest neighbor (KNN) and artificial neural network (ANN). KNN and ANN will use the cross-validation technique to choose the optimal number of neighbors and the best network respectively. All three methods will be evaluated in terms of the confusion matrix.

The advantages of using the decision tree algorithm are that the results are easy to interpret. The algorithm can handle all data types. The disadvantage of using the decision tree algorithm is that the algorithms usually overfit data if no constraints on the tree are defined before the fit. Computationally it requires little effort, and indirect feature selection is performed.
The KNN algorithm always converges and can be effective yet computationally hard for larger data sets as the distance to all observations is calculated for each iteration. It is not possible to give an interpretation of the variables as the number of neighbors, K, is the only thing that varies.

The neural network is like the two other methods non-parametric method. It requires lots of data and computational power to train the network. The structure of a neural net can, as we shall see, be rather complex, hence solve complex problems. It tends to outperform other methods, but the interpretation of the parameters is quite difficult.

The test set, which is used to predict with is, approximately 1700 rows. For prediction, a function is created to accumulate the prediction into a single energy label unique for each house. This is done by assigning the given houses the energy label is chosen by a majority vote.

5.2.1 Decision tree

Decision trees are usually referred to as non-metric classifiers. The decision tree consists of three types of nodes.

- A **root node**, which is the first (top) node of the tree. This is usually the decision which completes the biggest split of the data. This node has no incoming edges.
- **Internal nodes**, which is the middle of the decision tree. This node has both incoming and outgoing edges.

- **Terminal** nodes are the last nodes in the tree. This type of node has only incoming edges and determine the specific energy label. [27]

The primary algorithm used in the decision tree is called Hunts algorithm. The algorithm uses the term "purity," to define the structure of the tree. [23]. The purity gain is defined as

$$\Delta = Gini(\text{parent}) - \sum_{\text{children j}} \frac{N_j}{N} Gini(\text{child}\, j) \tag{5.4}$$

Where $I(x)$ is the purity of the node, N_j is the number of elements assigned to the child node j and N is the total elements in the parent node. The Hunts algorithm tries to find the split that maximizes the gain ratio. The parent node is the node above the child node. Say we're looking at Figure 5.8 at the node [A $>$ 0]. The parent node to this is [B $<$ 5] and the child node to [B $<$ 5] is [A $>$ 0].

The decision trees can use different methods for obtaining highest purity. Our algorithm has been set to use the Gini split. The Gini impurity is a probabilistic measure of how probable an attribute or class is to be in the given branch.

$$\text{Gini}(j) = 1 - \sum_{i=0}^{c-1} [p(i|j)]^2 \tag{5.5}$$

where $p(i|j)$ is the probability of j belonging to the class i in the given node j and c is the number of classes.
An example of a split using the Gini purity measure is shown in Figure 5.7. By asking a set of "questions," the purity gain is calculated for each question. The split with the highest purity is selected, here, the split in the right tree is chosen.

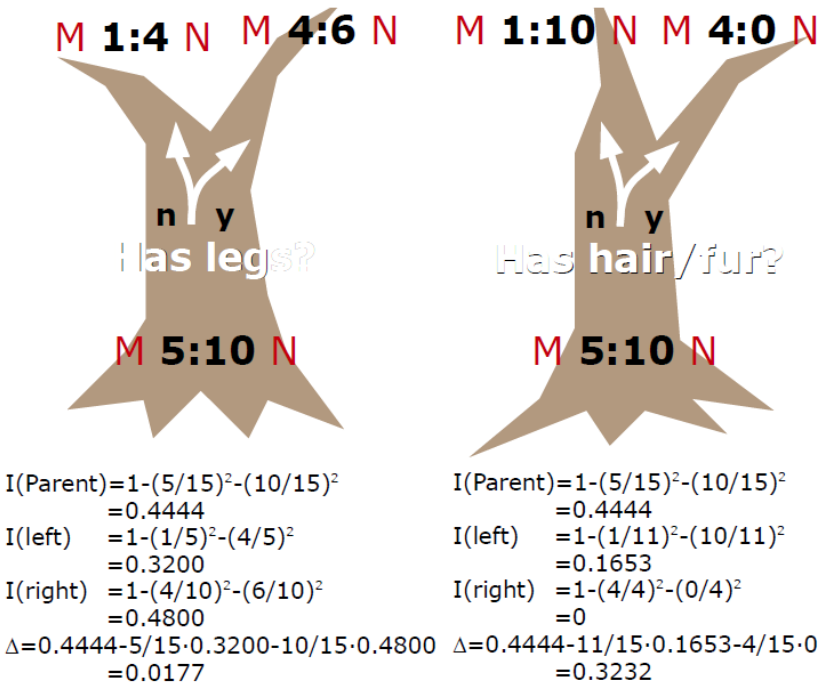

Figure 5.7: Example of a Gini split in a decision tree with 2 classes, M and N and 15 observations. It can be seen how the tree is splitted based on two different binary variables. Below the trees the purity gain is calculated.

5.2.1.1 How to use the decision tree

Say we have the following simple decision tree:

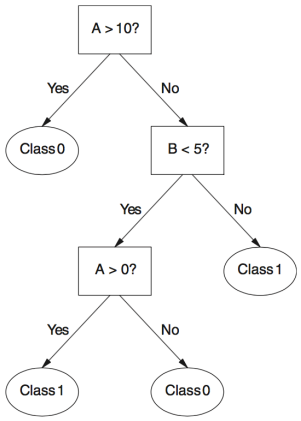

Figure 5.8: Example of a simple Decision tree

Lets say we have the following input data $[A, B] = [7, 3]$ and the tree in Figure 5.8. The algorithm starts at the root of the tree. Since $A = 7$ is not above 10, the algorithm goes down the "no" arrow, and because $B = 3$ is below 5, the algorithm goes to the left. This

continues until it terminates at *Class 1*.

The algorithm works the same way on a more complex tree, like in our examples, by answering yes/no questions all the way to the terminal nodes. The `predict`-command in R has been used to predict on the unseen test set.

5.2.1.2 Results

The tree is fitted with the R-function `rpart()` [28] with the given parameters `minsplit = 400`, `minbucket = 50` and `cp = 0.0001`. The given parameters could have been optimized, but for simplicity we have chosen this set of parameters that is able to be visualized and does not overfit to much.

`Minbucket` represents the minimum number of observations in any terminal node. `Minsplit` represents the minimum number of observations that exits in a node for it to be split. `Cp` is the complexity parameter representing the factor the split much decrease in the lack of fit test for the split to happen [28]. The following tree was fitted to our data:

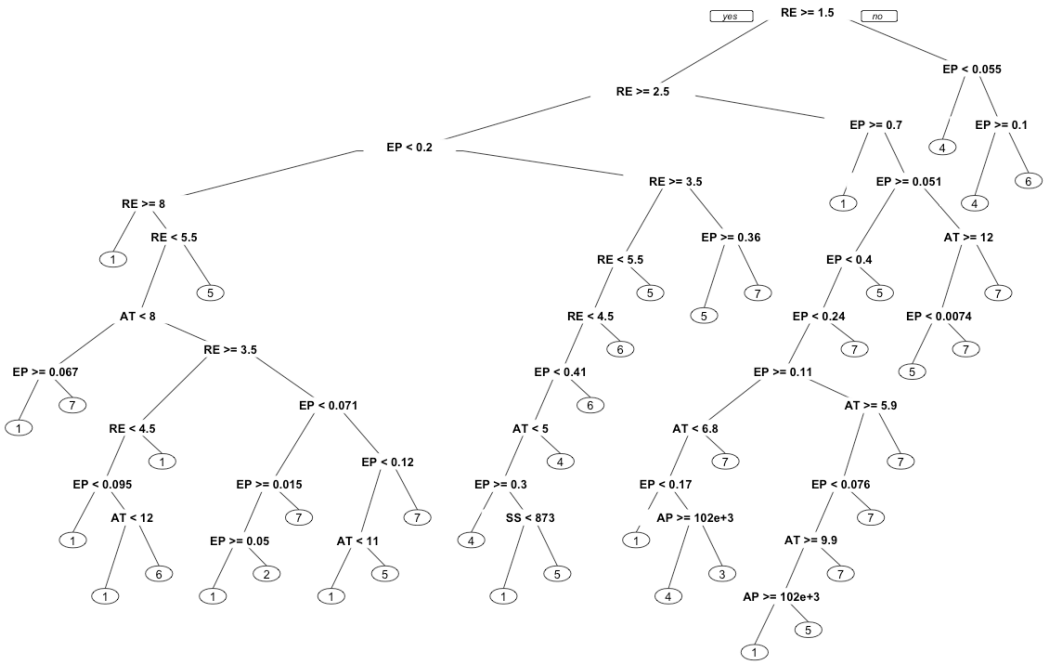

Figure 5.9: The fitted decision tree, AT represents `AirTemperature`, WS represents `WindSpeed`, AP represents `AtmosphericPressure`, SS represents `SurfaceSolarRadiationDownwards`, EP represents `EnergyPurchaseScaled` and RE represents `Residents`.

By looking at Figure 5.9 it can be seen that `Resident` is the root node splitting the tree into two big branches. The tree is dominated by `Resident` and `EnergyPurchaseScaled`.

This could indicate that these two variables are the most import to split the branches.

The decision tree is used to predict with the `predict`-command in R. From this the following confusion matrix is created.

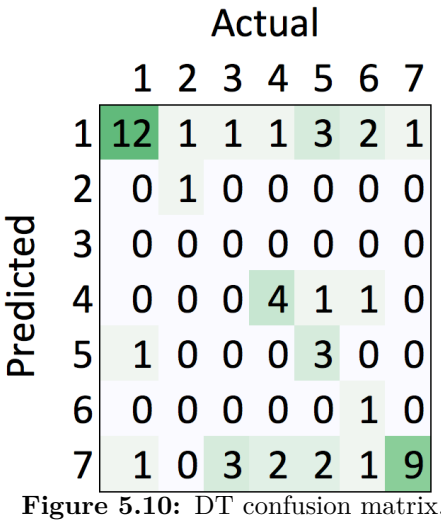

Figure 5.10: DT confusion matrix.

The following performance table has been set up:

Accuracy:	58.8 %						
Class:	1	2	3	4	5	6	7
Precision:	57.4 %	100 %	0 %	66.7 %	75.6 %	100 %	50 %
Recall:	85.7 %	50 %	0 %	57.1 %	33.3 %	20 %	90 %

Table 5.1: DT performance. 95 % CI: (0.4417, 0.7242)

It can be seen by looking at Table 5.1 that the algorithm is good a predicting, e.g., houses with an energy label A for which the actual energy label is A. However, the algorithm will predict many houses as A. This can also be seen from the performance measure where the precision for class 1 (A) is lower than the recall for class 1 (A).

The accuracy is calculated for how many times the algorithm predicts correctly. The accuracy does not take into account if it predicts e.g. the neighbor label, which also gives an indication on how the method performs. An algorithm has been implemented in which predicting one of each of the energy label neighbors are considered as a correct prediction. By doing this, the accuracy increases to 64.7 %. The code for doing this can be seen in A.2.2.

It is possible to see that the algorithm is not stable due to the large confidence interval on the accuracy. The accuracy might be as good as 3/4 but can also predict incorrect half the times. A narrower confidence interval is usually wanted as it is a sign of small variance of the estimates, but this was not possible to obtain when fitting this model. Other distance measures can be considered to improve the accuracy but was not examined.

5.2.2 K-nearest neighbors

The saying *"If it walks like a duck, quacks like a duck, and looks like a duck, then it is probably a duck"* represents the justification for the nearest neighbor algorithm, [27]. KNN is a distance measurement method, where K represents the number of neighbors the data points are compared to for any new instance. In Figure 5.11 a visualization of the KNN method is shown.

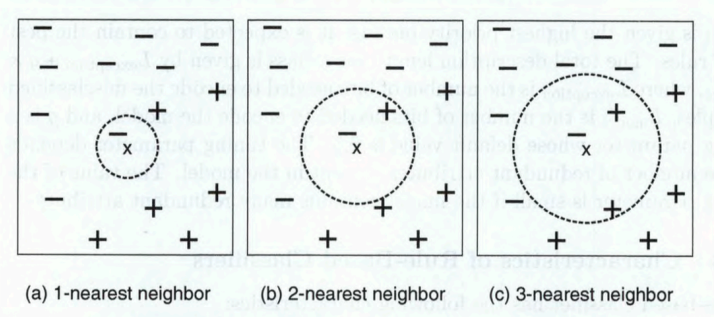

Figure 5.11: The KNN method using 1-,2- and 3-nearest neighbors. The minus and plus sign represent two different classes. The cross shows the center at which the distance is measured [27] and the circle indicates the distance to the K'th neighbor. If the neighbors are equally represented by the k neighbors, the label with the lowest distance to the instance is applied. Reproduced from [27]

Figure 5.11 only represents the data in a 2-dimensional space. In our dataset, there are six attributes in which the data point lies in a 6-dimensional space. By looking at Figure 5.11 it is possible to see how the instance is classified differently from choosing either 1-, 2- or 3-nearest neighbors. The classification is determined by a majority voting.
Several distance measures can be used to calculate the distance to the centroid. The usual measure is the Euclidean which is defined as:

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{5.6}$$

In terms of finding the optimal number of neighbors, a 10-fold cross-validation is made. The errors are compared, and the number of neighbors of the algorithms generating the lowest error is chosen.

5.2.2.1 Results

Using the 10-fold cross-validation to determine the optimal number of neighbors we obtain the following Error vs neighbors.

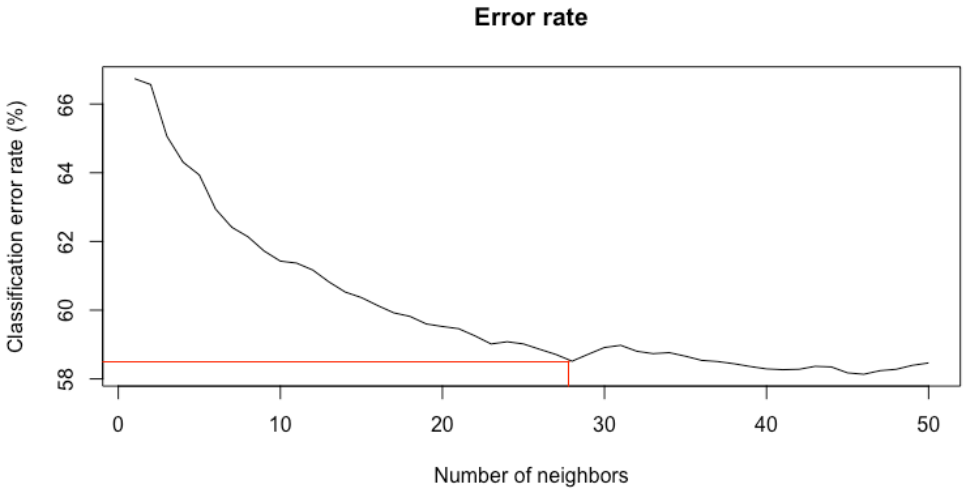

Figure 5.12: Error rate as a function of number of neighbors using euclidean distance.

By looking at Figure 5.12 it can be seen that the error rate drops as more and more neighbors are included. The error rate seems to increase a bit after 28 neighbors and then drops again later. To find the optimal number of neighbors, it is desired to find the lowest number of neighbors in which the error rate converges. In this, it is possible to avoid over- and under-fitting. 28 neighbors will be included in classifying the unseen data set. When using K = 28, the following confusion matrix and performance table was set up from the predictions:

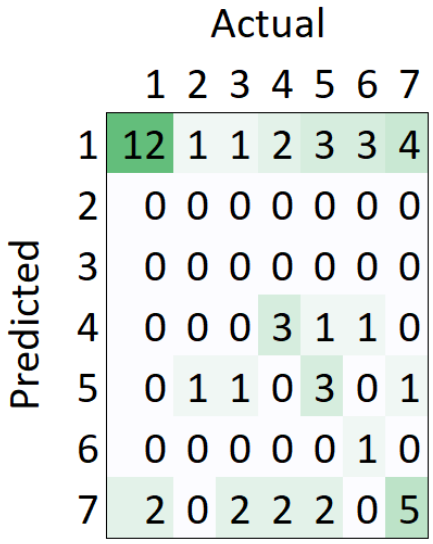

Figure 5.13: KNN confusion matrix with K = 28.

Accuracy:	47.0 %						
Class:	1	2	3	4	5	6	7
Precision:	44.4 %	0 %	0 %	60.0 %	50.0 %	100.0 %	38.5 %
Recall:	85.7 %	0 %	0 %	42.8 %	33.3 %	20.0 %	60.0 %

Table 5.2: KNN performance. 95 %: CI (0.3293-0.6154)

By looking at Table 5.2 it can be seen that the recall for energy label A is higher than the precision for energy label A. This means that the algorithm is great at predicting almost all of the actual energy label A classes as actual label A, but this also means that the algorithm predicts many houses to be A's even though the actual class of the given house was different from A. When extending the algorithm to also count adjacent neighbors as a correctly classified observation, the overall accuracy increased to 50.9 %.

It can be seen that the error rate at 28 neighbors is approximately 59 %, but the accuracy is 47 %. This is due to the majority vote. The results are done for multiple rows within a house. To obtain one single energy label prediction for each house the most frequent predicted energy label was chosen for this house. This might influence the overall error rate. However, it was assumed that the optimal number of neighbors also was optimal when aggregating to one unique label.

5.2.3 Artificial neural networks

The artificial neural network is inspired by the network which is found in the human brain. The human brain is built by synapses, neurons, axons, membranes, etc. These nodes communicate through electrical signals through quick impulses. All these signals determine the activity within the neural network in our brains.
By setting artificial nodes to represent the neurons, it is possible to generate an artificial network with similar functions as in the brain.
In machine learning, similar neural networks are trained and optimized by changing the number of hidden units, biases and weights. An example of a simple neural network can be seen in Figure 5.14.

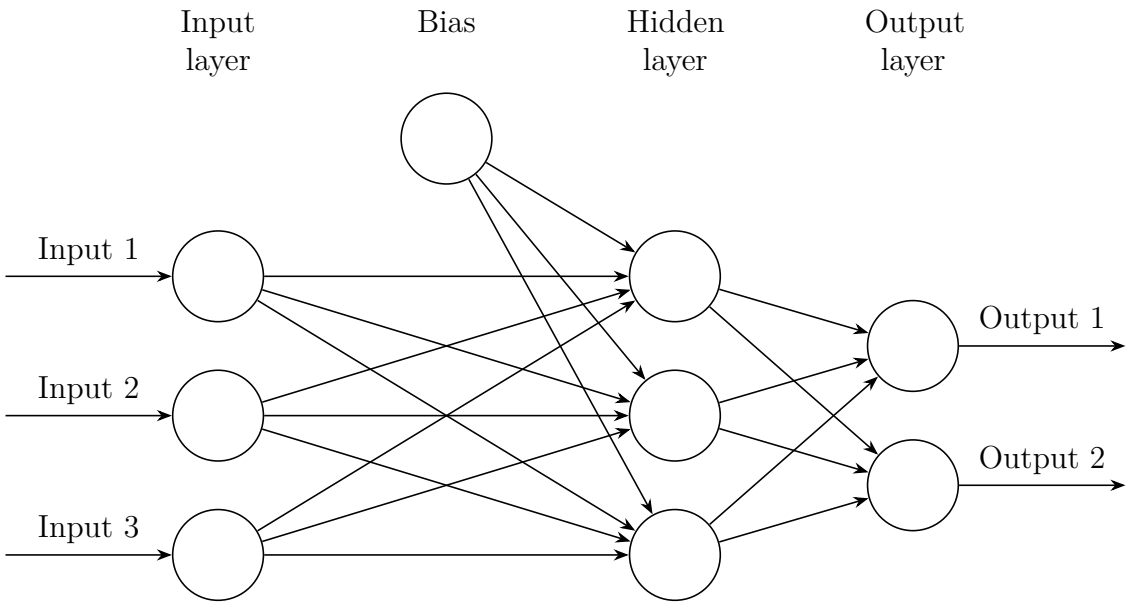

Figure 5.14: Simple neural network, with 1 hidden layer.

The simplest output signal from an ANN is the output between 0 and 1. The output represents a given probability of belonging to a given class. The input value is multiplied by the weight. Each hidden node receives n (number of inputs) weighted signals which are added together. The value in the hidden layer is denoted as

$$H_i = \sum_{i=1}^{n} w_i x_i \tag{5.7}$$

where w_i is the current weight for the specific x_i signal. H_i denotes the input value in the hidden node. Usually, neural networks are non-linear in the sense that within the hidden layer (H_i) a non-linear transformation is applied to the given values. Inside the hidden layer the Sigmoid activation function is applied:

$$f(z) = \frac{1}{1 + \exp^{-z}} \tag{5.8}$$

The Sigmoid function returns a number between -1 and 1.
To choose the optimal number of nodes in the hidden layer, we base our decision on the following quote *"the optimal size of the hidden layer is usually between the size of the input and size of the output layers"* - [19].

The number of input nodes is 6, and the number of output nodes is 7. It is not possible to have 6.5 nodes in the hidden layer so the mean sum of squared errors are compared and the lowest chosen. The MSSE is calculated using the RSS from Equation 4.2 as follows:

$$\text{RSS} = \sum_{i=1}^{n} (y_{test} - y_{est})^2$$
$$\text{MSSE} = \frac{\text{RSS}}{\text{N}}$$

where y_{est} are the estimated values of the output from the ANN, y_{test} are the original values of the test set, and N is the size of the training set.
The MSSE for six nodes in the hidden layer is 6.935, and the MSSE for seven nodes in the hidden layer is 6.877. From this, it is concluded that seven nodes are used in the neural network.

Bias nodes can be explained in terms of the linear regression line. When fitting a regression line, the relationship between the dependent and the explanatory variables are defined as $y = ax + b$. By changing the intercept b the straight line is better a fitting the data. The Sigmoid function which transforms the data in the hidden layer can be shifted to the left or right without changing the steepness of the curve.

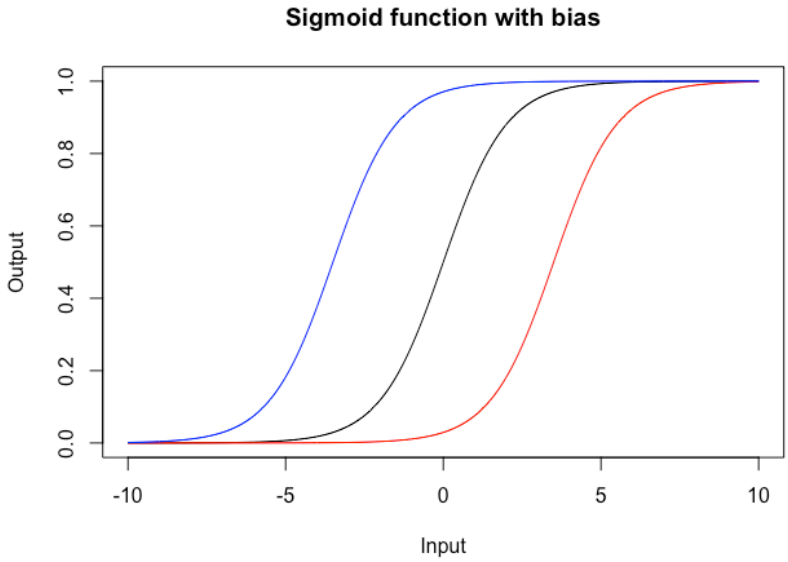

Figure 5.15: Bias visualized in the Sigmoid function. Adding the Bias term can shift the Sigmoid function.

From this, the Sigmoid function can be shifted to obtain a higher accuracy. The bias can be observed as B1 and B2 in Figure 5.16.

In order to calculate probabilities in the output node, the softmax function is used. The function transforms data transmitted from the hidden layer into probabilities which sums to one. The functions assigns probabilities of the given values to be a given class. The softmax function is defined as:

$$p_k = \frac{e^{x_i}}{\sum_{i=1}^{n} e^{x_i}} \tag{5.9}$$

where p_k is the probability of being in class k and x_i is the values coming from the hidden layer.

5.2.3.1 How to use a neural network

This section describes how a simple neural network roughly works. The algorithm also uses backpropagation and other more complicated algorithms, but this is only to understand how the different weights are used regarding Equation (5.7) and Equation (5.8). A further analysis and description on how backpropagation works can be found in [5]. Say we have estimated the following weights in Figure 5.14: $w_1 = 0.23, w_2 = 0.92, w_3 = -0.03, w_4 = 0.8, w_5 = 1.23, w_6 = 0.33, w_7 = -0.86, w_8 = 1.09, w_9 = -2.11, w_{10} = 0.05, w_{11} = 0.99, w_{12} = -1.14, w_{13} = 0.83, w_{14} = -0.31, w_{15} = -0.09$, the following input vector $[8.35, 2.92, -4.11]$ and the activation function used is the Sigmoid function. There are 3 bias weights $w_{16} = 5, w_{17} = 8.5, w_{17} = 5$.
The hidden layer is calculated using Equation (5.7).

$$\begin{aligned}
H_{1,\text{in}} &= 0.23 \cdot 8.35 + 0.92 \cdot 2.92 - 0.03 \cdot (-4.11) = 5.9632 \\
H_{2,\text{in}} &= 0.8 \cdot 8.35 + 1.23 \cdot 2.92 + 0.33 \cdot (-4.11) = 8.9153 \\
H_{3,\text{in}} &= -0.86 \cdot 8.35 + 1.09 \cdot 2.92 - 2.11 \cdot (-4.11) = 4.6739
\end{aligned}$$

The hidden layer is transformed using the Sigmoid function

$$\begin{aligned}
H_{1,\text{out}} &= \frac{1}{1 + \mathrm{e}^{-5.9632+1\cdot 5}} = 0.723762 \\
H_{2,\text{out}} &= \frac{1}{1 + \mathrm{e}^{-8.9153+1\cdot 8.5}} = 0.602358 \\
H_{3,\text{out}} &= \frac{1}{1 + \mathrm{e}^{-4.6739+1\cdot 5}} = 0.4191899
\end{aligned}$$

The hidden layer is then multiplied with the given weights to get the value for the two output nodes:

$$\begin{aligned}
O_1 &= 0.05 \cdot 0.723762 - 1.14 \cdot 0.602358 - 0.31 \cdot 0.4191899 = -0.87486624 \\
O_2 &= 0.99 \cdot 0.723762 + 0.83 \cdot 0.602358 - 0.09 \cdot 0.4191899 = 1.15134294
\end{aligned}$$

The softmax function from Equation (5.9) is applied

$$\begin{aligned}
O_1 &= \frac{\mathrm{e}^{-0.87486624}}{\mathrm{e}^{-0.87486624} + \mathrm{e}^{1.15134294}} = 0.1164785 \\
O_2 &= \frac{\mathrm{e}^{1.15134294}}{\mathrm{e}^{-0.87486624} + \mathrm{e}^{1.15134294}} = 0.8835215
\end{aligned}$$

The algorithm classifies the given data as class 2 due to a higher probability in O_2 compared to O_1.

5.2.3.2 Results

10-fold cross-validation is used to choose the optimal neural network. The network is each time trained 10 times in each fold with seven nodes in the hidden layer. The network with the lowest MSE is visualized in Figure 5.16. The neural net is fitted with the `nnet()`-command from [29].

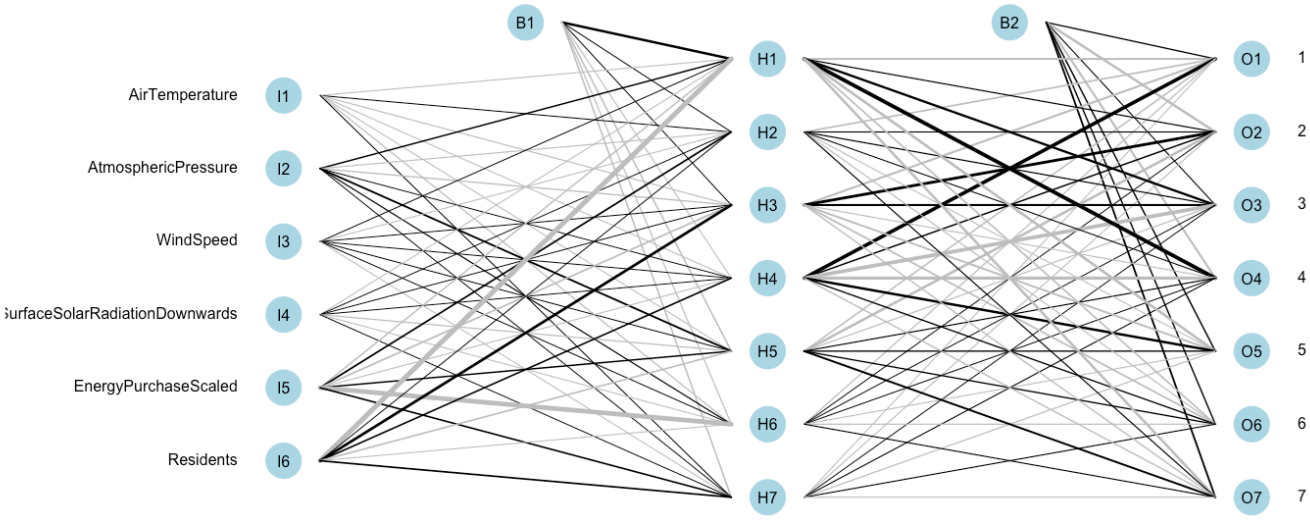

Figure 5.16: The fitted neural network containing the input layer with 6 attributes, one hidden layer with 7 nodes and the output layer with 7 nodes representing the 7 classes. Also, two bias nodes are included. The value of the weights can be seen in Table A.9.

It can be seen that some of the arrows are thicker than others corresponding to the size of the weights. The black weights are positive, and the gray weights are negative. When using the `predict`-command in R, it is possible to predict new energy labels using the optimal neural network in Figure 5.16. From this, the following confusion matrix and performance table has been set up

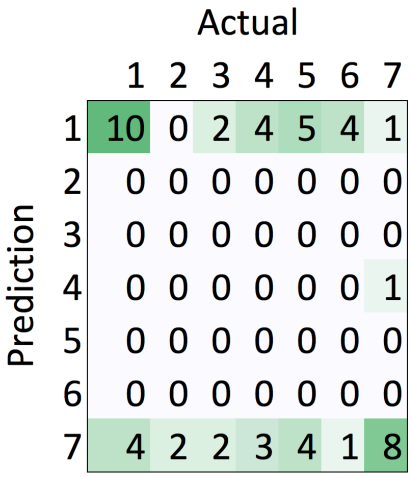

Figure 5.17: ANN confusion matrix.

Accuracy:	35.3 %						
Class:	1	2	3	4	5	6	7
Precision:	38.5 %	0 %	0 %	0 %	0 %	0 %	33.3 %
Recall:	71.4 %	0 %	0 %	0 %	0 %	0 %	80 %

Table 5.3: ANN performance. 95 % CI: (0.2243, 0.4993)

The recall is higher compared to the precision. The algorithm predicts almost all houses' energy labels as A or G but is predicting almost all houses whose energy label was A or G as their respective labels. It can be seen that the algorithm does not predict any houses as B, C, E, and F. This is probably caused by the lack of information in the given houses or because the network returns significant positive/negative values forcing the probabilities to end in A or G.

The algorithm was expanded to include neighbors increasing the accuracy to 37.3 %.

5.2.4 Results from the Logistic regression

Based on the observations made in Section 4.6.1, Figure 4.7 and 4.6 it is expected that the models will classify observations as A and G with a much higher probability than the rest of the labels. This is what is shown in Table 5.5 and Table 5.4. Out of the two methods, the multinomial logistic regression model is better at distinguishing the different labels, whereas the ordered logistic model was only able to predict A's and G's. It is possible to improve the results from the logistic regression by taking transformations into consideration or maybe use the cleaned data.

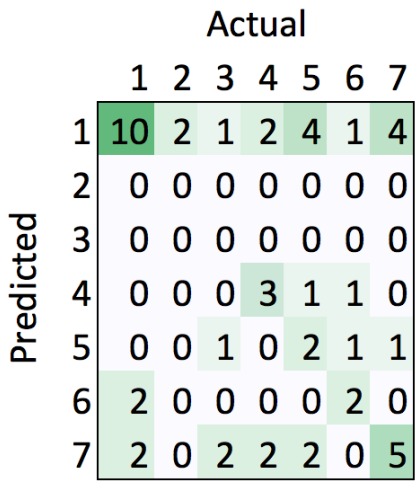

Figure 5.18: Confusion matrix of the multinomial regression model

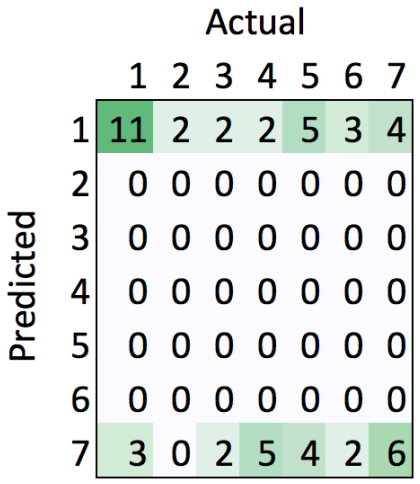

Figure 5.19: Confusion matrix of the ordered logistic regression model

Accuracy:	33.33%						
Class:	1	2	3	4	5	6	7
Precision:	41.7 %	0 %	0 %	60.0 %	40.0 %	50.0 %	38.5 %
Recall:	71.4 %	0 %	0 %	42.9 %	22.2 %	40.0 %	50.0%

Table 5.4: Ordered logistic regression performance. 95 % CI: (0.2076, 0.4792)

Accuracy:	43.13 %						
Class:	1	2	3	4	5	6	7
Precision:	37.9 %	0 %	0 %	0 %	0 %	0 %	38.5 %
Recall:	78.6 %	0 %	0 %	0 %	0 %	0 %	60.0 %

Table 5.5: Multinomial logistic regression performance. 95 % CI: (0.2935, 0.5775)

5.3 Collecting the results

The prediction for each of the five algorithms can be seen in Table 5.6.

Id	EL	DT	KNN	ANN	OL	ML	Id	EL	DT	KNN	ANN	OL	ML
1	5	1	1	1	1	1	27	6	7	1	1	7	5
2	3	7	7	1	7	7	28	5	1	1	1	1	1
3	3	1	1	7	1	1	29	1	1	1	1	1	1
4	1	1	1	1	1	1	30	3	7	5	1	1	5
5	2	2	5	7	1	1	31	7	7	7	7	1	7
6	7	7	7	1	7	7	32	7	7	7	7	7	7
7	6	6	6	1	1	6	33	1	1	1	7	7	6
8	3	7	7	7	7	7	34	5	5	5	1	7	1
9	4	4	4	1	7	4	35	6	4	4	1	1	4
10	1	1	1	1	1	1	36	1	1	1	7	7	1
11	1	1	1	1	1	1	37	4	7	7	1	1	7
12	4	4	4	7	7	4	38	1	1	7	1	7	7
13	1	7	7	1	7	7	39	4	1	1	1	7	1
14	7	7	1	7	7	1	40	1	1	1	7	1	1
15	4	4	4	1	7	4	41	7	7	7	7	7	7
16	7	7	7	7	7	7	42	7	7	1	4	1	1
17	5	4	4	7	7	4	43	1	1	1	1	1	1
18	5	5	5	7	7	5	44	6	1	1	1	7	6
19	4	7	7	7	1	7	45	2	1	1	7	1	1
20	7	7	1	7	7	1	46	1	1	1	1	1	1
21	6	1	1	7	1	1	47	4	4	1	7	1	1
22	5	1	1	7	1	1	48	5	7	7	7	7	7
23	5	7	7	1	1	7	49	7	1	1	7	1	1
24	7	7	5	7	7	5	50	1	1	1	7	1	1
25	1	1	1	1	1	1	51	1	5	1	1	1	6
26	5	5	5	1	1	5							

Table 5.6: Results from classification. It is possible to see the actual energy label and the predicted label. EL represent the original Energy label

By looking at Table 5.6 it can be seen that the neural network predicts differently compared to the other four methods, which predicts almost the same each time. Out of the five methods, it was only the decision tree that was able to predict any house as label B. In general; it seems as if the performance of the algorithms lacks due to the underlying distribution of the energy labels. Another thing to notice is the way some of the algorithms predict in the opposite end of each other. Take for example Id-5 which is a house labeled with a B. The predictions varies from A to G representing four different energy labels. The is no clear tendency for some of the houses when comparing the different methods. It can be seen that the ANN algorithm similar to the OL-algorithm predicts only As and Gs except for a few houses.

CHAPTER 6

Clustering

In the previous chapter, it has been shown how different methods can be used to predict energy labels through supervised learning, and the accuracy that can be obtained through knowing the real outcome. Unsupervised learning is now considered, where it is assumed that the energy labels are unknown. Like classification, similarities and patterns will be found, but this will be done by grouping observations in so-called clusters. Cluster analysis aims to find the observations that are the most similar to each other to create a grouping.

In this chapter cluster analysis will be used to cluster the data obtained through the linear regression analysis found in Table 4.1, but also through the BBR data that was not possible to include in the linear regression models. By doing this, it is feasible to see the possibilities of making a different type classification of the given data. Like classification, numerous clustering algorithms with different advantages and disadvantages exist. In this chapter, a few will be used to perform clustering to see how the houses might be grouped. An important part of clustering is to validate the number of clusters through cross-validation, but in this assignment, seven energy class labels are considered.

6.1 Methods

The data used will consist of the parameter estimates from Table 4.1 and the BBR data from Table A.1. Since Table 4.1 consists of models with a different number of parameters, it is decided to set the missing estimates to 0. Setting the estimates to 0 corresponds to the given term being insignificant and not contributing to the model. Since we will look at distances and similarity measures in cluster analysis, setting the missing estimates to 0 will not cause interference. We do not include the estimates of W_x as this is a discrete variable that interferes with some clustering algorithms, and in practice, it does not make sense to consider the direction of the wind as a variable to classify the houses energy label as we are more interested in the performance. It is also decided to remove the interaction between `AirTemperature` and `Windspeed`, as few houses included this term. The estimates to be included in the cluster analysis are then β_{uA}, β_{ws}, β_{gA}, β_{ap}, `Residents` and `BBR_YearOfConstruction`.

Two methods are considered, K-means and Gaussian Mixture models. In the next section, the methods are described. Some of the main differences are that K-means is non-

parametric using the position of each observation, and GMM is a parametric method optimizing the GMMS through variance and centers. In other words, K-means calculate distances, while GMM calculates “weighted” distances. K-means is based on Euclidean distance, where clusters are spherical. GMM can adjust itself to elliptically shaped clusters. When using both methods, it is assumed that all variables are continuous. Thus it is assumed that the variable `Residents` is continuous, which is not reasonable. It might seem like the GMM is better theoretically of the two methods. However, the K-means is easy to apply, and it converges very fast also for large data sets. Sometimes, the GMM is referred to as ”soft” clustering, and K-means is referred to as ”hard” clustering because in GMM a probability is assigned and in K-means a label is directly assigned. It is challenging to determine which method is the best, as we work with unsupervised learning and hence no ordering, however, we will try to compare it with the original 7 clusters and see if both methods can put each of the seven energy labels into their own clusters.

6.1.1 K-means

First, the number of initial K clusters are chosen by the user. Each data point is assigned to the closest centroid. The algorithm updates the position of each cluster. The algorithm repeats until no data point change cluster. The algorithm can be seen in a pseudo-code below.

Algorithm 2 Basic K-means algorithm [27]

1: Select K points as initial centroids
2: **repeat**
3: Form K clusters by assigning each point to its closest centroid
4: Update the position of each centroid
5: **until** Centroids do not change

To measure the performance of each cluster, the sum of squared error is calculated. The error represents the overall distance from all the data point assigned to a cluster to the centroid of the given cluster. Usually, cross-validation is used to find the optimal number of clusters, but due to our prior knowledge of the seven energy label categories, the number of clusters is fixed at 7. The SSE, which is desired as low as possible, is defined as

$$\text{SSE} = \sum_{i=1}^{K} \sum_{x \in C_i} dist(c_i, x)^2 \tag{6.1}$$

where K is the number of clusters, $x \in C_i$ is an object in the i^{th} cluster and $dist(c_i, x)$ is the Euclidean distance from the object x to the centroid. The centroid used Equation (6.1) is defined as

$$c_i = \frac{1}{m_i} \sum_{x \in C_i} x \tag{6.2}$$

where m_i is the number of data points in the i^{th} cluster.

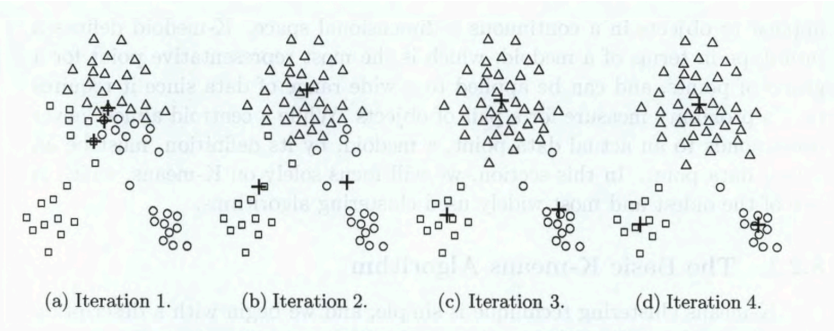

Figure 6.1: K-means, 4 iteration to find 3 clusters. Reproduced from [27].

By looking at Figure 6.1, it is possible to see how the algorithm iterates. In the first iteration, three centroids have been initialized in what looks like a cluster for itself. However, the algorithm can detect the 3 clusters in just four iterations. If the dimensionality in the data set is too large, the algorithm might not converge fast, and the optimal solution might not be found.

6.1.1.1 Results

To perform K-means clustering, the function `kmeans()` is used from [**nnet**]. The centers are randomly chosen amongst the data points, 20 random starts are used to find the best starting centers. Data has been normalized similarly to what is done the classification section. If no scaling is done, variables like `SolarSurfaceRadiationDownwards` and `AtmosphericPressure` will contribute more to the distance than other variables.

	A	B	C	D	E	F	G
1	0	0	1	0	1	0	1
2	7	2	0	0	0	0	0
3	0	0	0	0	1	1	0
4	0	0	0	4	0	1	0
5	0	0	1	2	5	1	2
6	0	0	1	0	0	0	1
7	0	0	0	0	2	0	1

Table 6.1: K-means clustering. The true classes A-G are divided into 7 new unordered clusters 1-7.

Table 6.1 shows how the houses were divided into 7 new clusters. There is no ordinal ranking between the clusters, so it is not possible to know if all the houses marked A also ended in class "A." What should be noted is that all houses marked A did fall into

the same cluster, meaning that these houses were similar to each other. Also, most of the D's and E's fell into the same cluster. The two houses in B also ended in the same cluster, however in the same cluster as A. If we look at the houses in G, it seems like they are randomly scattered between all of the clusters. Based on the numbers in Table 6.1, it seems as if there is no relationship between the houses within D, F, G.

Figure 6.2 shows the 7 cluster centers in a 2-dimensional space. It may seem like the clusters are very close to each other, but since we are only able to visualize the position of the clusters in 2d, they could in practice be 'far' from each other. It is possible to see which points were assigned to each cluster (circles). The true classes (points) should not be taken literally, but instead, it should be noted whether or not the true classes we are also in the same cluster.

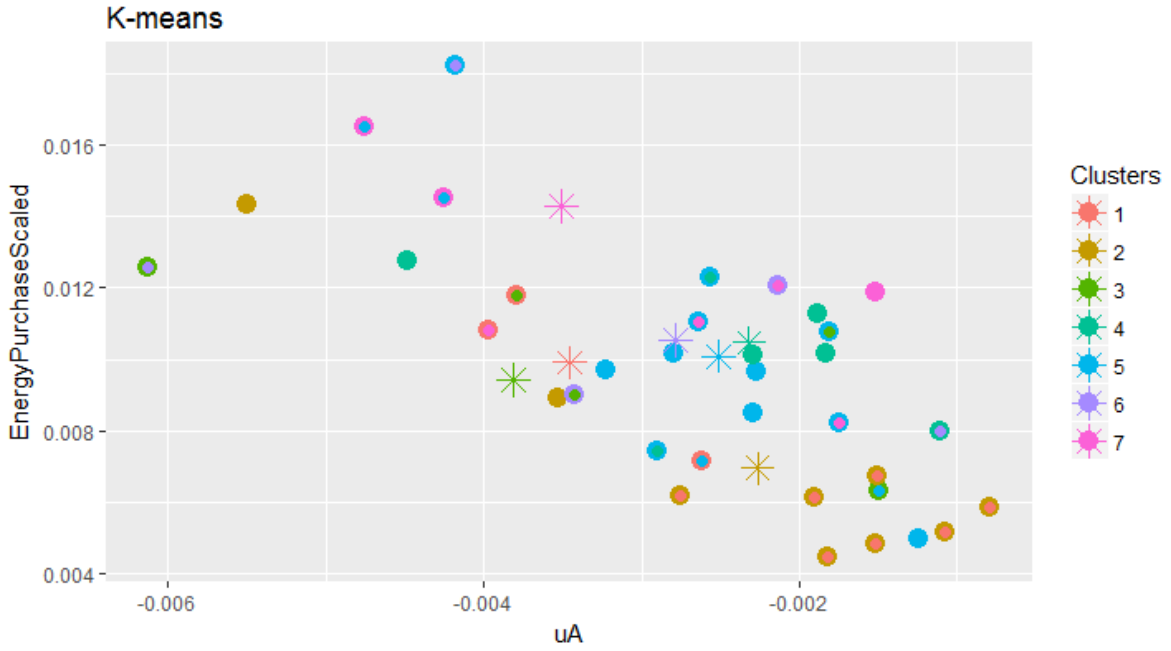

Figure 6.2: K-mean Clustering. The stars are the centers, the circles indicate cluster index, and the dots are the true classes.

6.1.2 Gaussian Mixture Models (GMMs)

Gaussian Mixture Models are an extension of the K-mean models. The algorithm uses probabilities to assign data to clusters. This probability is defined as:

$$p(x) = \sum_{i=1}^{K} w_k \mathcal{N}(x|\mu_{(k)}, \Sigma_{(k)}) \tag{6.3}$$

where K is the number of clusters, w_k is the size of the cluster, $\mu_{(k)}$ is the location of the cluster and $\Sigma_{(k)}$ is the covariance matrix of the cluster, meaning the shape of the

cluster and $p(x)$ is the probability of observing $\mathbf{x}$ at a given location. $\mathcal{N}$ represent the normal distribution.
where w_k is constrained to

$$\sum_{i=1}^{K} w_k = 1, \quad w_k \geq 0 \tag{6.4}$$

The GMM uses the EM (expectation maximization) algorithm to iterate. The EM algortihm can be explain in the following pseduo-code:

Algorithm 3 EM algorithm [27]

1: **Repeat**
2: 1. step - Expectation
3: For each object, calculate the probability of belonging to each distribution
4: 2. step - Maximization
5: For each probability distribution, estimate parameters using maximum
6: likelihood
7: **Until** the parameters do not change

More information about the EM algorithm can be found at [6].

6.1.2.1 Results

The function `mclust()` from the package `mclust` is used to fit the GMMs on the normalized data. The initial guesses for the EM algorithm are computed through hierarchical clustering, another fast non-parametric clustering method that relies on distances, chapter 8 in [27].

	A	B	C	D	E	F	G
1	0	0	1	6	5	2	2
2	0	2	0	0	0	0	0
3	0	0	1	0	1	0	1
4	7	0	0	0	0	0	0
5	0	0	0	0	2	0	1
6	0	0	1	0	0	0	1
7	0	0	0	0	1	1	0

Table 6.2: Gaussian Mixture Models. The true classes A-G are divided into 7 new unordered clusters 1-7.

This time, Table 6.2 shows a clearer clustering. The true classes A, B and D are all divided into their own clusters. The rest of the clustering seems to, as K-means, be randomly distributed. Figure 6.3 shows that the centers are a bit more separated, but still very similar to the K-means center positions.

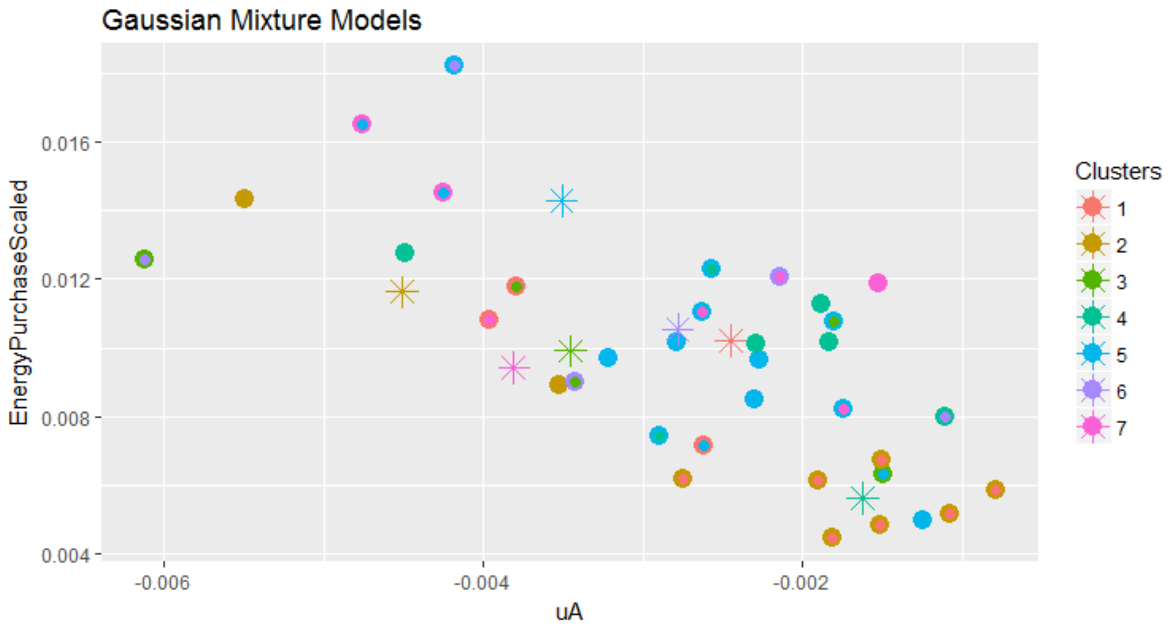

Figure 6.3: Gaussian Mixture Model. The stars are the centers, the outer circles indicate cluster index, and the dots are the true classes.

6.2 Comparison

Out of the two used methods, GMM was better at distinguishing between the real classes and slightly better at keeping the classes in the same cluster. Table 6.3 shows each house's true clustering along with the clusters from GMM and k-mean. As earlier mentioned, the true classes cannot be used for comparing, so it is the clusters in which the houses lie that should be compared. The two methods almost choose the same assignment of clusters, with the only difference being the placement of the actual class Bs, and that class D is not within the same cluster in K-means. As earlier mentioned, the GMM was able to divide 3 of the correct classes into three separate clusters, but as for the other clusters, it is mostly spread out from all other classes. Since all groups contain at least two houses, it is not expected that any of the houses can be considered as outliers, which typically will be in its own cluster, given this sample size.

It can be discussed whether or not including the BBR data would have any impact on predicting the energy labels, as it should not be the number of residents or the building year that affects the performance of the house. In Section 7.1, results from clustering analysis on a different data set, excluding BBR data, will also be presented. The approach is exactly as described in this chapter. In Chapter 8, we will connect the clustering with the regression analysis to see what similarities formed the clusters.

Id	EL	GMM	Kmean	Id	EL	GMM	Kmean
1	4	1	2	19	5	7	3
2	2	2	1	20	1	4	1
3	7	1	5	21	5	5	4
4	5	1	5	22	4	1	5
5	7	3	6	23	6	1	2
6	4	1	5	24	1	4	1
7	1	4	1	25	4	1	2
8	1	4	1	26	6	3	6
9	5	1	5	27	3	1	5
10	2	2	1	28	5	1	5
11	4	1	2	29	1	4	1
12	5	5	4	30	7	1	5
13	1	4	1	31	1	4	1
14	3	3	6	32	5	3	6
15	4	1	2	33	7	6	7
16	1	4	1	34	5	1	5
17	6	1	5	35	6	7	3
18	3	6	7				

Table 6.3: Results of the cluster analysis. It is possible to see the energy labels and the results of the cluster analysis for the Gaussian Mixture Models and K-means algorithm.

CHAPTER 7

Comparing the results

Up until know, many different algorithms and methods with different purposes have been used, and many results have been obtained. In this chapter, a summary of the key results and observations will be presented. Furthermore, the connection between regression, classification, and clustering will be discussed. In Chapter 8 the utilization of the results and suggestions for improvements will be discussed.

7.1 Linear regression and clustering

The multiple linear regression analysis showed that the prediction models of all houses were valid, as they fulfilled the assumptions listed in Section 4.1. Hence, the estimates found can be considered to be valid. Also, the simple linear regression models were for most models valid as well. The multiple linear regression analysis is, preferred as information from several variables are used. Using the estimates in Table 4.1 and comparing them to the rest of the houses, it is possible to see where pitfalls lie. This is limited to the data given.

	A	B	C	D	E	F	G
1	8	2	0	0	0	0	0
2	0	0	0	4	0	1	0
3	0	0	0	0	1	1	0
4	0	0	0	0	2	0	0
5	0	0	1	2	4	1	2
6	0	0	1	0	1	1	1
7	0	0	1	0	0	0	1

Table 7.1: K-means

	A	B	C	D	E	F	G
1	0	0	1	6	4	2	2
2	0	2	0	0	0	0	0
3	0	0	1	0	1	1	1
4	8	0	0	0	0	0	0
5	0	0	0	0	2	0	0
6	0	0	1	0	0	0	1
7	0	0	0	0	1	1	0

Table 7.2: GMM

Table 7.3: Clustering results with BBR data. Each row corresponds to a cluster. The parameters included are: `BBR_YearOfConstruction`, `Residents`, β_{uA}, β_{gA}, β_{ws} β_{ap} and the mean value of EPs.

When applying the estimates found in in the regression analysis, Table 4.1, and the BBR data A.1, the cluster analysis resulted in Table 7.3. At first sight, the clustering analysis seemed to be able to separate the true classes into their own clusters, preferring GMM over k-means. After inspecting the clusters further, it can be concluded that the division

is solely based on the variable `BBR_YearOfConstruction`. If only including this variable in the cluster analysis the results obtained yields the same clusters as in Table 7.3. The reason why the GMM and K-means in Table 7.3 are slightly different is because of the position of the centroids. In reality, newer houses are a lot more likely to have a better energy label compared to older houses, because of building regulations. As seen in Table 2.3 the division of YOC is very distinct

The cluster analysis is reapplied to the estimates only, using the same method as described in section 6. Table 7.5 shows the clustering when excluding the BBR data.

	A	B	C	D	E	F	G
1	3	1	3	5	4	0	2
2	0	0	0	1	0	0	0
3	3	0	0	0	1	1	0
4	0	0	0	0	1	0	0
5	0	1	0	0	1	1	0
6	0	0	0	0	2	1	2
7	1	0	0	0	0	0	1

Table 7.4: K-means

	A	B	C	D	E	F	G
1	3	0	0	3	3	0	2
2	1	1	3	2	2	0	2
3	0	0	0	0	1	2	1
4	3	0	0	0	1	1	0
5	0	1	0	0	1	0	0
6	0	0	0	1	0	0	0
7	0	0	0	0	1	0	0

Table 7.5: GMM

Table 7.6: Clustering results without BBR data. Each row corresponds to a cluster. The parameters included are: $\mu, \beta_{\text{uA}},\ \beta_{\text{gA}},\ \beta_{\text{ws}}\ \beta_{\text{ap}}$ and the mean value of EPs.

This time, the clusters formed are quite different from each other when comparing the two methods. It seems as if K-means form 1 large groups, and divides the rest of the observations into tiny clusters. The division of the true classes is less firm in the data set without BBR information. The clusters include combinations of labels from several different classes. Since the classes are ordinal, groups including both As and Gs, or other combinations of non-adjacent classes, is very conflicting. The statement of `EnergyLabel` being ordered seems to be rejected.

Id	EL	GMM_{BBR}	$\text{Kmean}_{\text{BBR}}$	GMM	Kmean	Id	EL	GMM_{BBR}	Kmean_{BBR}	GMM	Kmean
1	4	1	4	1	6	19	1	4	2	1	6
2	2	2	2	2	6	20	5	5	7	5	1
3	7	1	5	1	6	21	4	1	5	1	6
4	5	1	5	2	3	22	6	1	4	4	2
5	7	3	1	2	3	23	1	4	2	1	6
6	4	1	5	2	6	24	4	1	4	1	6
7	1	4	2	4	2	25	3	1	5	2	6
8	5	1	5	1	6	26	5	1	5	1	6
9	2	2	2	5	1	27	1	4	2	4	2
10	4	1	4	6	7	28	7	1	5	2	3
11	5	5	7	3	3	29	1	4	2	4	2
12	1	4	2	1	6	30	5	3	1	7	5
13	3	3	1	2	6	31	5	1	5	4	2
14	4	1	4	2	6	32	7	5	7	1	6
15	1	4	2	2	4	33	7	6	6	3	4
16	6	1	5	3	1	34	5	1	5	2	6
17	3	6	6	2	6	35	6	7	3	3	3
18	5	7	3	1	6	0	0	0	0	0	0

Table 7.7: Results of the cluster analysis including/excluding BBR information. EL represents the original energy label

By investigating the clusters returned by the GMM seen in Table 7.7 and comparing them to the estimates in Table 4.1, the following key features were discovered:

- Cluster 1: Includes low values of β_{uA}
  (Temperature sensitive)
- Cluster 2: Includes significant β_{ws} estimates and high β_{uA} values
  (Wind exposed and temperature sensitive)
- Cluster 3: Includes extremes
- Cluster 4: Includes low values of β_{uA}, β_{gA} and $\overline{EPs}$
  (Low exposure low baseline consumption)
- Cluster 5: Includes high values β_{uA} and β_{gA}
  (High exposure)
- Cluster 6: Includes high intercept
  (High baseline consumption)
- Cluster 7: Includes negative intercept
  (Low baseline consumption)

Some of the features are stronger bound than others. For example, there were lower β_{uA} values than the ones in Cluster 1. However, one of the other key features had a stronger similarity. Cluster 3 seems to contain a mix of extreme values as it includes the house with the highest β_{uA}, the house with the highest β_{gA}, the house with the lowest β_{uA} and the house with the highest mean EPs value. The estimates of β_{ap} do not seem to have any impact on the clustering, as they are randomly scattered between all clusters. It would be interesting to look at the variance of EPs instead of the mean, as the mean can be the same for a house with high or low variance.

7.2 Logistic regression

After several indicators of non-ordinal energy labels from the explorative analysis, classification, and clustering, the logistic regression analysis shows that reducing the multinomial logistic regression model into an ordered regression model was not significant. This was followed by a likelihood ratio test and the fulfillment of the parallel slopes assumption. Even though the ordered logistic regression model was valid, the models suffered from the distribution of the response.

Method	Accuracy
Ordered logistic regression	33.3 %
Multinomial logistic regression	43.3%

Table 7.8: Accuracy for the two methods

The classification accuracy was lower compared to some of the other algorithms. The models would probably benefit from model selection, as many of the p-values seen in A.8 were insignificant. Surprisingly there is a huge difference in the significant variables when comparing the ordered and multinomial model

7.3 Classification

All the models were able to predict almost all As and Gs correctly but had issues in classifying Bs and Cs. It is clear that the Decision Tree was the best method out of the five tried. This is not surprising as this is the method out of the three that requires the least "well-behaved" data. The neural network had one of the lowest accuracy, which can be explained by the lack of variation in the data. A neural network requires a sufficient amount of variation in the data. Take for example `EnergyLabel = B` where only houses with 3 and four persons live in the house. For the algorithm to work a larger variation properly within the number of residents within `Energylabel = B` would be acquired. Also, one of the advantages of ANN is that it can describe complex non-linear relationships, but the regression analysis showed that the data was mostly linear. Hence, it can be expected that the logistic regression models should do well. However, it requires sufficient variation in the data as well.

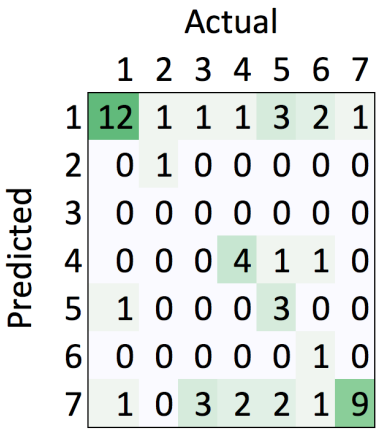

Figure 7.1: DT

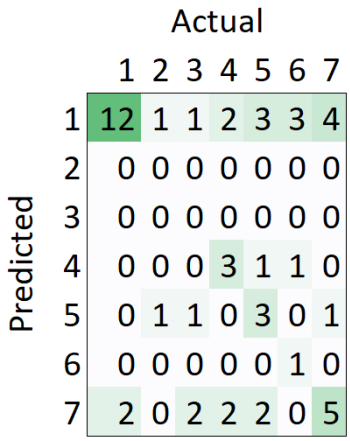

Figure 7.2: KNN

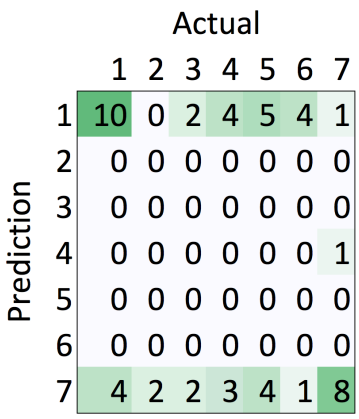

Figure 7.3: ANN

The algorithms compute their predictions based on patterns within each energy label. A reason why the accuracy is not higher in the three methods might be the lack of variation within the distribution of classes represented in the given data set.

Method	Accuracy
Decision tree	58.8 %
K-nearest neighbor	47.0 %
Artificial neural network	35.3 %

Table 7.9: Accuracy for the three methods

It could be proposed to make a naive model in which the model predicted all data points as A. If this were done an accuracy of 24.1 % would occur. It could also be proposed to make a naive model in which the model predicted all data points as Gs. If this were done, an accuracy of 20.2 % would occur. Compared to this benchmark it can be seen that the algorithms except ANN predict with an accuracy of approximately a factor 2.5 for each method.
However, if a model in which the model predicted all data points as either A or G (interpreted as a single class), an accuracy of 44.3 % would occur. By looking at this benchmark, it can be seen that the accuracy is almost as high as the three others. From this, it looks like the models were able to distinguish As from Gs, but based on the confusion matrices the middle section was more blurry. This could be another justification for saying that our data is not distributed ”well” enough overall energy labels. The ANN algorithm would be overruled by the naive model if it predicted solely on As and Gs, while the two other algorithms still would be better in accuracy than the naive model.

CHAPTER 8

Discussion

8.1 Application of the results

If SEAS-NVE is to ever implement the algorithms for classifying houses, they should use the classification algorithms to determine the energy labels.The cluster analysis could be used to divide the houses into groups with similar features. When keeping it separate, it is possible to classify the labels, followed by a comparison of other houses similar to the given house. This way the users can see how similar houses perform. The difficulty with the combination of classification and clustering lies in the fact that all houses are individual and they can have more in common with houses that have the same features than houses within the same energy label.

Since only 35 houses with five attributes, without BBR data, were used for the cluster analysis and 7 clusters were pre-determined, the sample size was simply too small. Our analysis did show that it was possible to determine what features made the clusters. If the cluster analysis only serves the purpose of minimizing the error, a different number of clusters might be found with more distinguished differences. The choice of 7 levels was chosen to accommodate the levels of energy labels, but through cross-validation, the optimal number of clusters can easily be determined. From the regression part, the marginal effect of the different variables can be inspected by investigating the non-transformed variables, although there were signs of violations of the linear regression assumptions.

The prediction accuracy of the classification methods reached at maximum almost 60 % without counting neighbor predictions. Given the fact that seven classes were considered, it can see as quite sufficient, since a random guess results in an accuracy of $100/7 = 14.3\%$. It could have been considered to use 5-fold or any number of folds cross-validation. There is a bit uncertainty whether or not the high accuracy was due to the distribution of the energy labels. *"The performance of model-based estimates may be determined more by the number of an event rather than the total sample size"* [20], seems to perfectly explain the issues with both the logistic regression and classification part. All classification algorithms appear to have problems with exactly this. It would be interesting to re-run the algorithms with a data set, containing a larger variation in energy labels, to see the possibilities of assigning energy labels to houses with a higher accuracy.

8.2 Future aspects and extensions

As has been discussed so far, the sample size was a bit of an obstacle. If we look past this, the following suggests possible extensions for future work.

Penalized errors:
During the classification analysis, errors were equally penalized no matter the distance to the true class. One might consider weighting the errors, so being one class away contributed less to the error than i.e. houses two classes apart. This would be interesting due to the fact the algorithm gives and an indication whether the given house was predicted in the lower or higher end of the energy label spectrum.

Secondary heating type:
Including the secondary heating, type would be appropriate, as it may explain the patterns of the energy consumption. The secondary heating type was included in the BBR data, however, it was also too user specific. Adding this variable to the analysis would have resulted in the same issues that were discovered when including the building year. Including the variable would be on the condition that more data was available.

Precipation:
To get more information about the performance of the house, including a variable measuring precipitation would be interesting. It is common that houses with a damp environment, use more heat to make up for this. Including this variable can give the users more specific advise on where their pitfalls may lie.

Include the prior distribution:
As the distribution of the energy labels is known, [13], it is possible to include the prior distribution in Bayesian statistics (naive Bayes classification). If the prior is included, it would be possible to overcome models that tend to classify based on the number of events due to the sample size. In some areas, the distribution of energy labels is different from other areas. These different distribution could be taken into account in a more location-specific model.

Human behaviour:
It might be interesting to include more and more information on the people living in the house. What type of persons is living in the house might have a huge influence on their consumption. It could be proposed to include an input for the user in the app in which the user typed their baseline consumption (indoor temperature) to get an indication what the given person preferences are.

Separate models:
If a sufficient amount of data is available, it would be interesting to create separate models for each energy label. Hence, the comparison not with similar houses, but houses with the same energy label would be available. This can give the users another kind of comparison, than the one the cluster analysis provided.

CHAPTER 9

Conclusion

Bringing down the electricity bill and sparing the environment is something that concerns most people. For any house owner, knowing the energy label and performance of his/hers house can be a costly affair. The possibility of providing the users of Watts app a free guiding data-driven energy label is within reach. It was discovered that determining the energy label was best when including the year of construction, but the variable was left out as the performance of the house is investigated. Advice on where the pitfalls lie can be provided to some extent. Through regression analysis, it is possible to create models for each house with significant variables. The estimates of the variables can say something about the meteorological variables, hence give indicators of pitfalls. It was possible to group these estimates through clustering methods, in which key features was possible to extract and interpret. Giving houses a different kind of label can hence be provided. In this, a more general unordered ”label” can be given, describing the house performance. These features can help SEAS-NVE target specific users that are in need of advice.

Despite the lacking results obtained more research and data is needed to improve the accuracy of the labeling. If SEAS-NVE chooses to continue with the methods in this thesis, they should determine at which level of precision they are satisfied. Other possible extensions to the analysis have been proposed to improve the current results.

APPENDIX A

Appendix

A.1 Tables and plots

Id	Id	Residents	Year	EnergyLabel	StartDate	EndDate
1	636109273549864000	1	1981	D	2016-05-27	2016-09-24
2	636123227017892000	4	2003	B	2016-05-17	2016-10-04
3	636123613882942000	3	1976	G	2016-05-17	2016-09-29
4	636125073579127000	4	1975	E	2016-05-04	2016-09-24
5	636125489145044000	2	1950	G	2016-05-17	2016-09-24
6	636125491697329000	4	1972	D	2016-05-17	2016-10-04
7	636125635995978000	4	2014	A	2016-06-06	2016-09-14
8	636125864856504000	4	2016	A	2016-04-13	2016-08-20
9	636128356912679000	3	1971	E	2016-05-04	2016-10-04
10	636129883033486000	3	2003	B	2016-04-13	2016-10-04
11	636129892259820000	4	1982	D	2016-04-13	2016-09-30
12	636131605290084000	3	1925	E	2016-05-22	2016-09-24
13	636131795718900000	3	2011	A	2016-05-10	2016-09-14
14	636131936879050000	4	1946	C	2016-05-27	2016-09-25
15	636133663320120000	1	1981	D	2016-06-01	2016-10-04
16	636137740469289000	4	2010	A	2016-06-06	2016-09-14
17	636142383326354000	4	1973	F	2016-05-17	2016-10-04
18	636145634513274000	2	1870	C	2016-04-24	2016-09-29
19	636146259359494000	2	1892	E	2016-05-27	2016-10-04
20	636148809161288000	3	2011	A	2016-04-13	2016-09-24
21	636149012017354000	2	1916	E	2016-05-17	2016-09-25
22	636149076337899000	2	1977	D	2016-04-24	2016-10-04
23	636149203254479000	3	1979	F	2016-05-10	2016-09-24
24	636149295254777000	4	2010	A	2016-05-27	2016-09-14
25	636150707728923000	2	1988	D	2016-05-04	2016-09-24
26	636151564359982000	5	1959	F	2016-04-13	2016-10-11
27	636155987669325000	3	1964	C	2016-06-06	2016-10-04
28	636166284612245000	6	1976	E	2016-05-04	2016-09-29
29	636178608145571000	3	2011	A	2016-05-10	2016-09-14
30	636183369260198000	3	1976	G	2016-05-10	2016-10-15
31	636196343071176000	5	2013	A	2016-05-04	2016-09-14
32	636198405280220000	4	1955	E	2016-04-29	2016-10-03
33	636226036099149000	2	1877	G	2016-04-13	2016-09-24
34	636235232632072000	1	1976	E	2016-06-06	2016-10-04
35	636241688833619000	1	1896	F	2016-04-29	2016-10-19

Table A.1: Table of the cleaned regression data. StartDate refers to the first day of the removed data, and EndDate refers to the last day of the removed data

Id	Id	Residents	Year	EnergyLabel	StartDate	EndDate
1	636103267026721000	2	1932	G	2015-12-31	2017-01-01
2	636109273549864000	1	1981	D	2015-12-31	2017-01-01
3	636116997638683000	10	2011	A	2015-12-31	2017-01-01
4	636123227017892000	4	2003	B	2015-12-31	2017-01-01
5	636123613882942000	3	1976	G	2015-12-31	2017-01-01
6	636125073579127000	4	1975	E	2015-12-31	2017-01-01
7	636125489145044000	2	1950	G	2015-12-31	2017-01-01
8	636125491697329000	4	1972	D	2015-12-31	2017-01-01
9	636125635995978000	4	2014	A	2015-12-31	2017-01-01
10	636125864856504000	4	2016	A	2015-12-31	2017-01-01
11	636128356912679000	3	1971	E	2015-12-31	2017-01-01
12	636129883033486000	3	2003	B	2015-12-31	2017-01-01
13	636129892259820000	4	1982	D	2015-12-31	2017-01-01
14	636131605290084000	3	1925	E	2015-12-31	2017-01-01
15	636131795718900000	3	2011	A	2015-12-31	2017-01-01
16	636131936879050000	4	1946	C	2015-12-31	2017-01-01
17	636133588659802000	4	1974	F	2015-12-31	2017-01-01
18	636133663320120000	1	1981	D	2015-12-31	2017-01-01
19	636137740469289000	4	2010	A	2015-12-31	2017-01-01
20	636142383326354000	4	1973	F	2015-12-31	2017-01-01
21	636144662913484000	3	2012	G	2015-12-31	2017-01-01
22	636145634513274000	2	1870	C	2015-12-31	2017-01-01
23	636146259359494000	2	1892	E	2015-12-31	2017-01-01
24	636148809161288000	3	2011	A	2015-12-31	2017-01-01
25	636149012017354000	2	1916	E	2015-12-31	2017-01-01
26	636149076337899000	2	1977	D	2015-12-31	2017-01-01
27	636149203254479000	3	1979	F	2015-12-31	2017-01-01
28	636149295254777000	4	2010	A	2015-12-31	2017-01-01
29	636150707728923000	2	1988	D	2015-12-31	2017-01-01
30	636151395049333000	1	1981	D	2015-12-31	2017-01-01
31	636151564359982000	5	1959	F	2015-12-31	2017-01-01
32	636153898519167000	4	1859	G	2015-12-31	2017-01-01
33	636155987669325000	3	1964	C	2015-12-31	2017-01-01
34	636158320190564000	2	2016	A	2015-12-31	2017-01-01
35	636159210620210000	3	2016	A	2016-11-11	2017-01-01
36	636166284612245000	6	1976	E	2015-12-31	2017-01-01
37	636171269122310000	3	2017	A	2016-11-28	2017-01-01
38	636175941243776000	2	2016	A	2016-02-15	2017-01-01
39	636178608145571000	3	2011	A	2015-12-31	2017-01-01
40	636183369260198000	3	1976	G	2015-12-31	2017-01-01
41	636196343071176000	5	2013	A	2015-12-31	2017-01-01
42	636197607378811000	2	2007	C	2015-12-31	2017-01-01
43	636198405280220000	4	1955	E	2015-12-31	2017-01-01
44	636203222276620000	2	2012	G	2016-01-02	2017-01-01
45	636204384845907000	3	1972	E	2015-12-31	2017-01-01
46	636225928078269000	2	1920	G	2016-01-01	2017-01-01
47	636226036099149000	2	1877	G	2015-12-31	2017-01-01
48	636234590331952000	4	2014	A	2016-03-31	2017-01-01
49	636235232632072000	1	1976	E	2015-12-31	2017-01-01
50	636241688833619000	1	1896	F	2015-12-31	2017-01-01
51	636242247345996000	5	1837	G	2015-12-31	2017-01-01

Table A.2: Table of the cleaned classification data. StartDate refers to first date in the data, and EndDate refers to the last date

Id	Intercept	uA	Id	intercept	uA
1	0.11636	-0.00250	19	0.07481	-0.00202
2	0.10953	-0.00388	20	0.14685	-0.00485
3	0.10053	-0.00231	21	0.10008	-0.00348
4	0.11024	-0.00300	22	0.09498	-0.00120
5	0.12192	-0.00391	23	0.08976	-0.00192
6	0.12350	-0.00317	24	0.11387	-0.00254
7	0.07988	-0.00140	25	0.12066	-0.00277
8	0.10791	-0.00250	26	0.10382	-0.00294
9	0.14044	-0.00574	27	0.08970	-0.00227
10	0.12859	-0.00464	28	0.11705	-0.00262
11	0.14097	-0.00431	29	0.07819	-0.00181
12	0.09159	-0.00314	30	0.09503	-0.00285
13	0.13004	-0.00427	31	0.07589	-0.00099
14	0.11398	-0.00288	32	0.11389	-0.00180
15	0.09056	-0.00278	33	0.12684	-0.00440
16	0.15403	-0.00455	34	0.11222	-0.00277
17	0.10798	-0.00337	35	0.12332	-0.00319
18	0.08791	-0.00208			

Table A.3: Estimates of the simple linear regression models with a square root transformation

Id	sign-simple	KS-simple	signmulti	KS-multi	Id	sign-simple	KS-simple	sign-multi	KS-multi
1	0	1	0	0	19	0	1	0	1
2	0	1	0	1	20	0	1	0	1
3	0	1	0	1	21	0	1	0	1
4	0	1	0	0	22	0	1	0	0
5	0	1	0	1	23	0	1	0	1
6	0	1	0	1	24	1	1	0	1
7	0	1	0	1	25	0	1	0	1
8	0	1	0	0	26	0	0	0	1
9	0	1	0	1	27	0	1	0	1
10	0	1	0	1	28	0	1	0	1
11	0	1	0	1	29	0	1	0	1
12	0	1	0	1	30	0	1	0	1
13	0	1	0	1	31	0	1	0	0
14	0	1	0	1	32	1	1	0	1
15	0	0	0	1	33	0	0	0	1
16	0	1	0	1	34	0	1	0	1
17	0	1	0	1	35	0	1	0	1
18	0	1	0	0	36	0	0	0	0

Table A.4: Sign- and Kolmogorov-Smirnov test of the errors of the linear regression analysis. 1 denotes the house passed the test, 0 denotes failed.

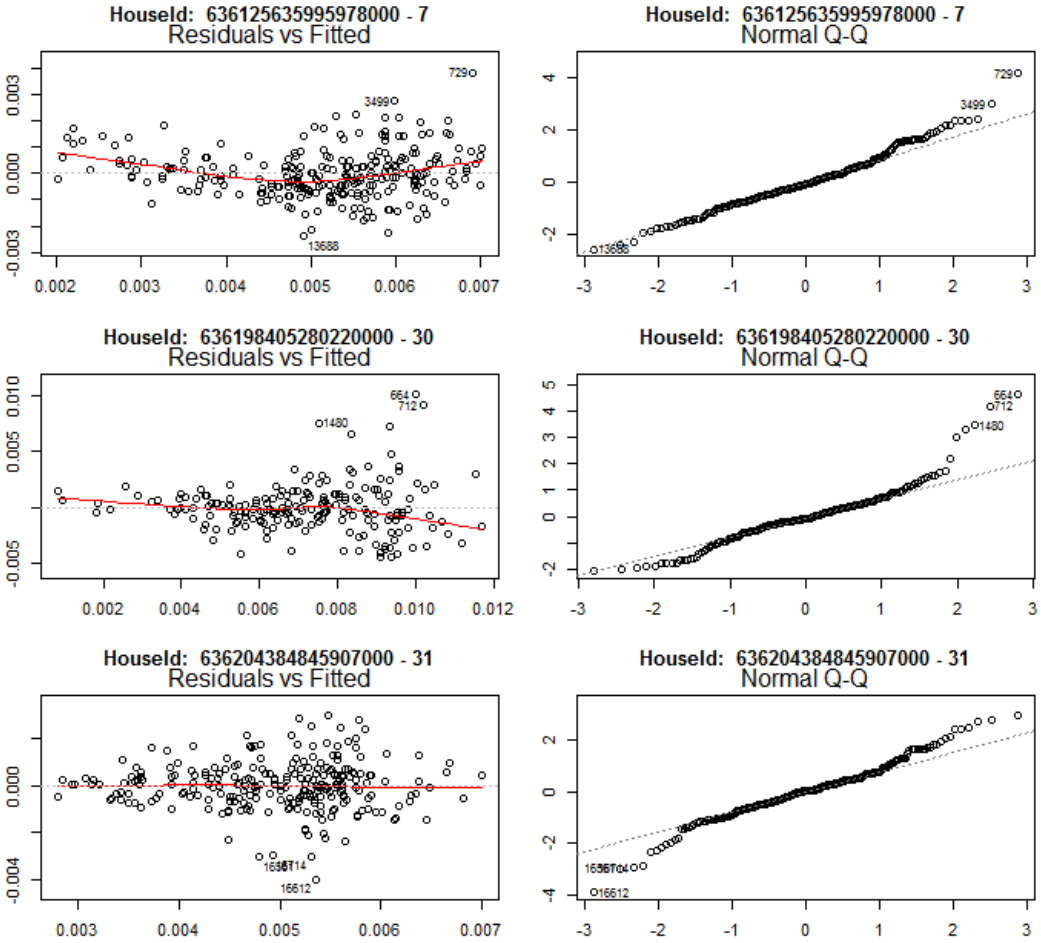

Figure A.1: Selected residual plots of the non-transformed multiple linear regression

EnergyLabel	B	C	D	E	F	G
(Intercept)	5.82	13.85	29.32	33.62	19.84	11.65
AirTemperature	0.31	0.16	0.91	0.50	0.79	0.05
EnergyPurchaseScaled	7.06	6.28	8.27	5.15	8.39	-0.93
Residents2	-12.68	-2.00	-16.66	-15.63	-29.84	2.96
Residents3	0.04	-3.34	-31.91	-15.87	-17.05	1.66
Residents4	-0.15	-3.60	-17.58	-16.58	-16.55	0.09
Residents5	-15.07	-19.22	-32.03	-36.97	-15.64	1.42
Residents6	1.15	0.27	-15.16	0.65	-12.08	3.12
Residents10	-13.63	-17.17	-29.41	-33.25	-30.33	-15.36
WindSpeed	-0.30	-0.37	-0.82	-0.91	0.20	-0.67
SurfaceSolarRadiationDownwards	0.68	0.65	-0.27	-0.10	0.58	-0.81
AtmosphericPressure	-7.95	-12.52	-13.29	-18.43	-5.21	-13.37

Table A.5: Estimates of the multinominal logistic regression model, where label A is the reference class

	Estimate	Std. Error	t-value	P-value
AirTemperature	0.13196717	7.774010e-02	1.697543e+00	8.959402e-02
EnergyPurchaseScaled	0.04958203	3.475468e-01	1.426629e-01	8.865564e-01
Residents2	0.41353024	4.781144e-02	8.649190e+00	5.186619e-18
Residents3	-0.18435502	4.805559e-02	-3.836287e+00	1.249085e-04
Residents4	-0.75246427	4.679195e-02	-1.608106e+01	3.463866e-58
Residents5	0.28779671	7.722800e-02	3.726585e+00	1.940918e-04
Residents6	0.28006343	8.840355e-02	3.168011e+00	1.534856e-03
Residents10	-37.15182868	2.943649e-13	-1.262101e+14	0.000000e+00
WindSpeed	-0.23928837	1.132932e-01	-2.112116e+00	3.467654e-02
SS	-0.35644372	6.898821e-02	-5.166734e+00	2.382200e-07
AP	-5.97316671	1.615851e+00	-3.696608e+00	2.184995e-04
1|2	-7.39094495	1.584860e+00	-4.663470e+00	3.109217e-06
2|3	-7.14936483	1.584773e+00	-4.511286e+00	6.443573e-06
3|4	-6.71605475	1.584616e+00	-4.238286e+00	2.252325e-05
4|5	-6.08171189	1.584426e+00	-3.838431e+00	1.238228e-04
5|6	-5.23224348	1.584344e+00	-3.302468e+00	9.583811e-04
6|7	-4.62644475	1.584353e+00	-2.920085e+00	3.499357e-03

Table A.6: Estimates of the ordered logistic regression model

Class	RE1	RE2	RE3	RE4	RE5	RE6	RE10
1	-7.39	-7.52	-7.52	-7.52	-7.52	-7.52	-7.52
2	-7.15	-7.20	-7.20	-7.20	-7.20	-7.20	-7.20
3	-6.72	-7.13	-6.53	-5.96	-7.00	-7.00	30.44
4	-6.08	-5.84	-5.84	-5.84	-5.84	-5.84	-5.84
5	-5.23	-4.88	-4.88	-4.88	-4.88	-4.88	-4.88
6	-4.63	1.35	1.35	1.35	1.35	1.35	1.35

Table A.7: Log odds from the ordered logistic regression model for each class and each level of `Residents`

	μ	β_{uA}	β_{EPs}	$\beta_{RE=2}$	$\beta_{RE=3}$	$\beta_{RE=4}$	$\beta_{RE=5}$	$\beta_{RE=6}$	$\beta_{RE=10}$	β_{ws}	β_{SS}	β_{ap}
B	0.640	0.161	0.000	0.702	0.997	0.990	0.000	0.729	0.000	0.389	0.001	0.106
C	0.676	0.355	0.000	0.952	0.919	0.913	0.000	0.996	0.709	0.161	0.000	0.001
D	0.015	0.000	0.000	0.154	0.558	0.132	0.032	0.838	0.398	0.001	0.066	0.000
E	0.005	0.000	0.000	0.181	0.174	0.156	0.000	0.984	0.000	0.000	0.455	0.000
F	0.101	0.000	0.000	0.512	0.144	0.156	0.181	0.767	0.736	0.419	0.000	0.145
G	0.439	0.712	0.236	0.842	0.911	0.995	0.924	0.965	0.000	0.001	0.000	0.000

Table A.8: P-values of the multinominal model. Class A is considered as the reference groups

Nodes	B1	I1	I2	I3	I4	I5	I6	
H1	50.87	-1.43	17.97	1.32	-0.13	-11.80	-131.04	
H2	2.32	1.79	-2.90	-0.44	0.34	26.81	1.00	
H3	2.36	-2.96	-9.05	0.50	0.69	-22.28	59.14	
H4	-1.12	-0.26	0.25	0.15	0.14	-1.38	21.86	
H5	-14.55	-0.53	30.54	1.69	-5.50	18.73	-18.42	
H6	-4.78	-5.97	8.52	1.54	-0.71	-120.72	-0.02	
H7	-14.57	2.74	0.81	-0.24	0.81	23.22	25.35	
Nodes	B2	H1	H2	H3	H4	H5	H6	H7
O1	12.62	-31.99	-20.97	-34.29	87.75	-14.51	-6.32	-0.09
O2	-46.04	8.80	5.67	63.59	18.03	-49.19	1.20	-2.17
O3	5.53	45.78	9.16	33.88	-88.91	-4.74	1.05	0.79
O4	-33.93	94.53	4.33	-2.90	-70.50	7.68	1.96	5.06
O5	17.04	-51.82	-1.52	-28.42	51.53	14.63	-1.39	-5.32
O6	15.59	-18.45	-3.75	-21.99	13.70	13.61	-3.39	5.61
O7	29.13	-48.71	8.14	-10.32	-10.61	34.23	5.94	-4.58

Table A.9: Weights of fitted neural network. The table shows the weights between each node

A.2 R-code

A.2.1 Machine learning preamble

```
library(cvTools) #for machine learning code (lr)
library(R.matlab) #for machine learning code (lr)
library(rpart) #for decission tree
require(caret) #decission tree matrix
library(e1071) #decission tree matrix
library(FNN) #KNN
library(nnet)
library(rpart.plot)

set.seed(123) #seed for sampling
watts.data3 <- watts.data[sample(nrow(watts.data)),]
N <- length(watts.data3[,1])
N_train <- round(N/10)*9
N_test <- N - N_train
y <- watts.data3$EnergyLabelNumber
X <- watts.data3[,-2]

X <- X[,c("Id","AirTemperature", "AtmosphericPressure", "WindSpeed",
          "SurfaceSolarRadiationDownwards", "EnergyPurchaseScaled",
          "Residents")]
X_train <- X[c(1:N_train),]
# make sure that the response is interpreted as a categorical variable
y_train <- y[c(1:N_train)]
y_train <- as.factor(y_train)
X_test <- na.omit(X[c(N_train+1:N),])
y_test <- na.omit(y[c(N_train+1:N)])
attributeNames <- as.vector(colnames(X))
X_traindf <- data.frame(X_train)
X_testdf <- data.frame(X_test)
colnames(X_testdf) <- attributeNames
colnames(X_traindf) <- attributeNames
X_traindf$Residents <- as.numeric(X_traindf$Residents)
X_testdf$Residents <- as.numeric(X_testdf$Residents)

X$Residents <- as.numeric(X$Residents)

#Remove unused data frames
rm(watts.data3)
```

A.2.2 Functions

```
#Link to confusionMatrix
#http://stackoverflow.com/questions/25497398/how-to-construct-the-confusion
#-matrix-for-a-multi-class-variable
```

```
#Link to theory behind multiclass confusionmatrix
#http://blog.revolutionanalytics.com/2016/03/com_class_eval_metrics_r.html

#Fix so confusionmatrix ignore different amount of levels
#http://stackoverflow.com/questions/19871043/r-package-caret-confusionmatrix-with
#-missing-categories

CM <- function(X_test, y_test, y_test_est, method, dt ) {

  y_test_est_bind <- matrix(data = NA,nrow = length(y_test_est),ncol = 2)
  for ( i in 1:length(y_test_est)){
    y_test_est_bind[i,1] <- as.character(X_test$Id[[i]])
    y_test_est_bind[i,2] <- as.numeric(y_test_est[[i]])
  }
  (y_test_est_bind <- as.data.frame(y_test_est_bind))

  y_test_bind <- matrix(data = NA,nrow = length(y_test),ncol = 2)
  for ( i in 1:length(y_test)){
    y_test_bind[i,1] <- as.character(X_test$Id[[i]])
    y_test_bind[i,2] <- as.numeric(y_test[i])
  }
  (y_test_bind <- as.data.frame(y_test_bind))

  if ( method == "dt"){
    n <- length(levels(y_test_bind$V1))
    estimates <- matrix(data = NA ,nrow = n, ncol = 3)
    for ( i in 1:n){
      estimates[i,1] <- levels(y_test_bind$V1)[i]
      estimates[i,2] <- names(which.max(table(y_test_est_bind$V2[
        which(y_test_est_bind$V1 == unique(y_test_est_bind$V1)[i])])))
      estimates[i,3] <- names(which.max(table(y_test_bind$V2[
        which(y_test_bind$V1 == unique(y_test_bind$V1)[i])])))
    }
    if ( dt == 1 ){
      diff <- as.numeric(estimates[,2])-as.numeric(estimates[,3])
      error.rate <- length(diff[which(abs(diff)>1)])/length(estimates[,3])
      accuracy<-1-error.rate
      return(paste('Accuracy when using neigboor as correct is:', round(accuracy,3)))
    }
    return(confusionMatrix(estimates[,2],estimates[,3]))
  }

  if ( method == "knn") {
    n <- length(levels(y_test_bind$V1))
    estimates <- matrix(data = NA ,nrow = n, ncol = 3)
    for ( i in 1:n){
      estimates[i,1] <- levels(y_test_bind$V1)[i]
      estimates[i,2] <- names(which.max(table(y_test_est_bind$V2[
        which(y_test_est_bind$V1 == unique(y_test_est_bind$V1)[i])])))
      estimates[i,3] <- names(which.max(table(y_test_bind$V2[
        which(y_test_bind$V1 == unique(y_test_bind$V1)[i])])))
```

```
    }
    u <- union(estimates[,3], estimates[,2])
    t <- table(factor(estimates[,3], u), factor(estimates[,2], u))
    return(confusionMatrix(t))
  }
}
```

```
model.select.linear= function(factors, data, starttime, endtime, Idnum,
scale_unscaled,transform, criteria){
#scale_unscaled: 1 = unscaled
#transform: 1 = log(), 2 = sqrt(), 3 =()^2, else = none
#criteria: 1 = t-test, 2 = AIC
  dataset = subset(data, data$Id == Idnum & (data$day <= starttime |
    data$day >= endtime))
  attach(dataset)
  EnergyPurchase_scaled = EnergyPurchase/SquareMeters
  if (scale_unscaled == 1){  #unScaled
    if (transform == 1){
      response = log(EnergyPurchase)
    }  else if(transform == 2) {
      response = sqrt(EnergyPurchase)
    } else if(transform == 3){
      response = EnergyPurchase^2
    } else {
      response = EnergyPurchase
    }
    model = lm(response ~ AirTemperature, data=dataset)
    scopes =  as.formula(paste("~", paste(factors, collapse="+")))

  } else {                    #scaled
    if (transform == 1){
      response = log(EnergyPurchase_scaled)
    }  else if(transform == 2) {
      response = sqrt(EnergyPurchase_scaled)
    } else if(transform == 3){
      response = EnergyPurchase_scaled^2
    } else {
      response = EnergyPurchase_scaled
    }
    model = lm(response ~ AirTemperature, data=dataset)
    scopes =as.formula(paste("~", paste(factors, collapse="+")))
  }

  while(criteria==1){
    x = add1(model, scopes, test="F")
    F_value = min(x[,6], na.rm=T)
    var = rownames(x[which.min(x[,6]),])
    if(F_value >= 0.051)
      break
    #   else if ( aic >= x[1,5])
    #      break
```

```
    formul <- as.formula(paste(".~.+",var))
    model = update(model,formul)
  }

  while(criteria==2){
    x = add1(model,scopes)
    aic = min(x[,4])
    var = rownames(x[which(x[,4]==min(x[,4])),])
    if(var == "<none>")
      break
    else if ( aic >= x[1,4])
      break
    formul <- as.formula(paste(".~.+",var))
    model = update(model,formul)
  }
  detach(dataset)
  return(model)
}
```

A.2.3 Load classification data

```
#### PACKAGES AND LOAD DATA ####
rm(list=ls())
library(readr) # for loading data
library(ggplot2) #for better plots
library(gridExtra) #for grid.arrange in ggplot2
library(plotrix) #for radial plot
library(fields) #for contourplot
library(GGally) #for gg correlation plot
library(scales)

watts <- read_delim("~/Google Drev/DTU/6. semester/Bachelor/Data/31.03.2017/
    wattsdata31.03.2017.txt", "\t",
    escape_double = FALSE, col_types = cols(Id = col_character(),
    X30 =col_skip(), X31 = col_skip(), X32 = col_skip(), X33 = col_skip()),
    trim_ws = TRUE)

#Remove houses with squaremeters = 0.
watts <- subset(watts, SquareMeters > 0)

watts$EnergyPurchase <- gsub(",", ".", watts$EnergyPurchase)
watts$EnergyPurchase <- as.numeric(watts$EnergyPurchase)
watts$EnergyPurchaseScaled <- watts$EnergyPurchase/watts$SquareMeters
watts$Id <- as.factor(watts$Id)

#Change heatingtype and housetype to number
watts$BBR_HeatingType[which(watts$BBR_HeatingType=="NOTSPECIFIED")] <- 1
watts$BBR_HeatingType[which(watts$BBR_HeatingType=="Electricity")] <- 2
watts$BBR_HeatingType[which(watts$BBR_HeatingType=="Wood")] <- 3
```

```
watts$BBR_HeatingType[which(watts$BBR_HeatingType=="WoodPellets")] <- 4
watts$BBR_HeatingType[which(watts$BBR_HeatingType=="NaturalGas")] <- 5
watts$BBR_HeatingType[which(watts$BBR_HeatingType=="FuelGasOil")] <- 6
watts$BBR_HeatingType <- as.numeric(watts$BBR_HeatingType)

watts$BBR_HouseType[which(watts$BBR_HouseType=="Holiday")] <- 1
watts$BBR_HouseType[which(watts$BBR_HouseType=="SingleFamily")] <- 2
watts$BBR_HouseType[which(watts$BBR_HouseType=="MultiFamily")] <- 3
watts$BBR_HouseType[which(watts$BBR_HouseType=="NonResidental")] <- 4
watts$BBR_HouseType <- as.numeric(watts$BBR_HouseType)

watts$WindDirectionDescription <- as.factor(watts$WindDirectionDescription)

#Fix data
watts$Id <- as.factor(watts$Id)
watts$day <- as.Date(substring(watts$StartTime, 1,10),"%d-%m-%Y")
watts$time <- substring(watts$StartTime, 12,13)

#Remove hot days

watts <- subset(watts, (watts$BBR_HouseType==3 | watts$BBR_HouseType==2) &
                  (EnergyPurchaseScaled>0.001 & EnergyPurchaseScaled<0.14))

#Remove NA's
watts1 <- na.omit(watts)


#### Acummulated weather data   ####
#Means
dt.means <- aggregate(cbind(AirTemperature,AtmosphericPressure, WindSpeed,
WindDirectionDegree, IsWeekDay, IsWeekend, IsHoliday, DefCode,
WeekDay, IsWeekDay, IsWeekend, PostalCode, BBR_HeatingType, BBR_HouseType,
HeightOverSeaLevel, Residents, SquareMeters, BBR_YearOfConstruction,
EnergyLabelNumber) ~ Id + day, data = watts1, FUN = mean , na.rm = TRUE)

dt.means <- dt.means[order(dt.means$Id),]
dt.means <- as.data.frame(dt.means)

#Sums
dt.sums <- aggregate(cbind(EnergyPurchase,SurfaceSolarRadiationDownwards,

TotalSkyDirectRadiationAtSurface) ~ Id + day, data = watts1, FUN = sum ,
    na.rm =TRUE)
dt.sums <- dt.sums[order(dt.sums$Id),]
dt.sums <- as.data.frame(dt.sums)

#Combine data
watts.data <- cbind(dt.means, dt.sums)

#Create scaled energypurchase
watts.data$EnergyPurchaseScaled <- watts.data$EnergyPurchase/watts.data$SquareMeters
```

```
#### Visualization of weather data ####
graphics.off()

#Remove unsued data frames
rm(watts1, dt.means, dt.sums, watts)
```

A.2.4 Load regression data

```
watts <- read_delim("C:/Users/Banki/Google Drev/Bachelor/Data/31.03.2017/
wattsdata31.03.2017.txt", "\t", escape_double = FALSE,
col_types = cols(Id = col_character(), X30 = col_skip(),
X31 = col_skip(), X32 = col_skip(), X33 = col_skip()), trim_ws = TRUE)

watts <- subset(watts, SquareMeters > 0)

watts$EnergyPurchase <- gsub(",", ".", watts$EnergyPurchase)
watts$EnergyPurchase <- as.numeric(watts$EnergyPurchase)
watts$EnergyPurchaseScaled <- watts$EnergyPurchase/watts$SquareMeters
watts$Id <- as.factor(watts$Id)

p = ggplot(watts, aes(x = EnergyLabelNumber, y= EnergyPurchaseScaled,
    group = EnergyLabel, fill = EnergyLabel)  )
P = p + geom_boxplot()+theme(legend.position='none') +
  scale_x_discrete(limit = c("A", "B", "C", "D", "E", "F", "G")) +
  ggtitle("Energy Consumption for original data") + xlab("EnergyLabel")
q = ggplot(watts, aes(y = Residents, x= EnergyLabel, group = as.factor(Residents),
    fill = as.factor(Residents)) )
Q = q + geom_boxplot()

x[,1] =as.factor(as.matrix(as.data.frame(watts$Residents)))
y=as.factor(as.matrix(as.data.frame(watts$EnergyLabelNumber)))


#Change heatingtype and housetype to number
watts$BBR_HeatingType[which(watts$BBR_HeatingType=="NOTSPECIFIED")] <- 1
watts$BBR_HeatingType[which(watts$BBR_HeatingType=="Electricity")] <- 2
watts$BBR_HeatingType[which(watts$BBR_HeatingType=="Wood")] <- 3
watts$BBR_HeatingType[which(watts$BBR_HeatingType=="WoodPellets")] <- 4
watts$BBR_HeatingType[which(watts$BBR_HeatingType=="NaturalGas")] <- 5
watts$BBR_HeatingType[which(watts$BBR_HeatingType=="FuelGasOil")] <- 6
watts$BBR_HeatingType <- as.numeric(watts$BBR_HeatingType)

watts$BBR_HouseType[which(watts$BBR_HouseType=="Holiday")] <- 1
watts$BBR_HouseType[which(watts$BBR_HouseType=="SingleFamily")] <- 2
watts$BBR_HouseType[which(watts$BBR_HouseType=="MultiFamily")] <- 3
watts$BBR_HouseType[which(watts$BBR_HouseType=="NonResidental")] <- 4
watts$BBR_HouseType <- as.numeric(watts$BBR_HouseType)

watts$Residents[which(watts$Residents >= 6)] <- 6
```

```
watts$Residents = as.factor(watts$Residents)
watts$WindDirectionDescription <- as.factor(watts$WindDirectionDescription)

#Fix data
watts$Id <- as.factor(watts$Id)
watts$day <- as.Date(substring(watts$StartTime, 1,10),"%d-%m-%Y")
watts$time <- substring(watts$StartTime, 12,13)

#Remove holidays
watts = watts[!c(watts$IsHoliday),]

#Remove hot days
watts <- subset(watts, (watts$BBR_HouseType==3 | watts$BBR_HouseType==2) &
    !(watts$Residents==0))

#Remove NA's
watts1 <- na.omit(watts)

#### Acummulated weather data ####
#Means
dt.means <- aggregate(cbind(AirTemperature,AtmosphericPressure, WindSpeed,
WindDirectionDegree, IsWeekDay, IsWeekend, IsHoliday, DefCode, WeekDay,
IsWeekDay, IsWeekend, PostalCode, BBR_HeatingType, BBR_HouseType,
HeightOverSeaLevel, Residents, SquareMeters, BBR_YearOfConstruction,
EnergyLabelNumber, EnergyPurchase, EnergyPurchaseScaled) ~
Id + day, data = watts1, FUN = mean , na.rm = TRUE)

#Remove hot days
dt.means <- subset(dt.means, (dt.means$BBR_HouseType==3 |
dt.means$BBR_HouseType==2) & !(dt.means$Residents==0))

dt.means$WindDirectionDescription[dt.means$WindDirectionDegree<22.5 |
    dt.means$WindDirectionDegree> 337.5] = "N"
dt.means$WindDirectionDescription[dt.means$WindDirectionDegree<337.5 &
    dt.means$WindDirectionDegree> 292.5] = "NV"
dt.means$WindDirectionDescription[dt.means$WindDirectionDegree<292.5 &
    dt.means$WindDirectionDegree> 247.5] = "V"
dt.means$WindDirectionDescription[dt.means$WindDirectionDegree<247.5 &
    dt.means$WindDirectionDegree> 202.5] = "SV"
dt.means$WindDirectionDescription[dt.means$WindDirectionDegree<202.5 &
    dt.means$WindDirectionDegree> 157.5] = "S"
dt.means$WindDirectionDescription[dt.means$WindDirectionDegree<157.5 &
    dt.means$WindDirectionDegree> 112.5] = "SE"
dt.means$WindDirectionDescription[dt.means$WindDirectionDegree<112.5 &
    dt.means$WindDirectionDegree> 67.5] = "E"
dt.means$WindDirectionDescription[dt.means$WindDirectionDegree<67.5 &
    dt.means$WindDirectionDegree> 22.5] = "NE"
dt.means$WindDirectionDescription <- as.factor(dt.means$WindDirectionDescription)

dt.means <- dt.means[order(dt.means$Id),]
dt.means <- as.data.frame(dt.means)
```

```
#Sums
dt.sums <- aggregate(cbind(SurfaceSolarRadiationDownwards,
    TotalSkyDirectRadiationAtSurface) ~ Id + day, data = watts1,
    FUN = sum , na.rm =TRUE)
dt.sums <- dt.sums[order(dt.sums$Id),]
dt.sums <- as.data.frame(dt.sums)

#Combine data
watts.data <- cbind(dt.means, dt.sums)

#Remove hot days
watts.data <- subset(watts.data, EnergyPurchaseScaled>0.001)

#Correct the type
watts.data$EnergyLabelNumber = as.factor(watts.data$EnergyLabelNumber)
watts.data$IsWeekDay = as.factor(watts.data$IsWeekDay)
watts.data$IsWeekend = as.factor(watts.data$IsWeekend)
watts.data$IsHoliday = as.factor(watts.data$IsHoliday)
watts.data$DefCode = as.factor(watts.data$DefCode)
watts.data$PostalCode = as.factor(watts.data$PostalCode)
watts.data$BBR_HeatingType = as.factor(watts.data$BBR_HeatingType)
watts.data$BBR_HouseType = as.factor(watts.data$BBR_HouseType)
watts.data$Residents = as.factor(watts.data$Residents)


#Remove NA's
watts.data <- na.omit(watts.data)

#Remove double ID
watts.data= watts.data[,-c(25,26)]

#Remove houses with less than 300 values####
uniq <- unique(watts.data$Id)
n.uniq <- length(uniq)
len = 0
for ( j in 1:n.uniq){
  data.subset <- subset(watts.data, Id == uniq[[j]])
  len[j] =  dim(data.subset)[1]
}
len
index  = which(len < 300)
ids = seq(1,n.uniq,1)[-index]
x=c()
for(i in ids){
x=  rbind(x,subset(watts.data, Id == uniq[i]))
}
n = length(unique(x$Id))
watts.data =x
uniq <- unique(watts.data$Id)

####Export relevant data####
dyn<- read_delim("~/DTU 15-16/bach/dyn4.txt", " ", escape_double = FALSE,
```

```
    col_types = cols(`11 11` = col_number()), trim_ws = TRUE)
dyn  = cbind(dyn[,1], dyn[,2])
Idnr = as.character(0) ;  start=as.Date("16-01-01"); slut= as.Date("16-01-01")
for ( i in 1:n){
data.subset <- subset(watts.data, Id == uniq[[i]])
  uniq.day <- unique(data.subset$day)
  n.length <- length(uniq.day)
  if (dyn[i,2] == 0){
  dyn[i,2] = n.length
}
  if (dyn[i,1] == 1){
    dyn[i,1] =2
  }
  start[i] = uniq.day[dyn[i,1]]
  slut[i] = uniq.day[dyn[i,2]]
  if(start[i] <= "2016-01-10"){
    start[i] = "2000-01-01"
  }
  if(slut[i] >= "2016-12-20"){
    slut[i] = "2000-01-01"
  }
}

#Export data
dynamic_days = cbind(as.character(uniq), as.character(start), as.character(slut))
dynamic_days = dynamic_days[-c(1,3,16,28,30,35),]
colnames(dynamic_days) = c("Idnr","start","slut")

##### remove unused data####
rm(watts, dt.sums, dt.means, watts1, x,x1 ,data.subset, dyn, dynamic_days)
```

A.2.5 Exploration and visualization of data

```
####LOAD DATA ####
rm(list=ls())
source("Loaddata1.R") #load data and accumulate etc.
source("Functions.R") #functions
#### Visualization ####

#Plot of all data #
#EnergyPurchase AirTemperature
dt.full <- aggregate(cbind(EnergyPurchase, AirTemperature) ~
                       day, data = watts.data, FUN = mean , na.rm =TRUE)
plot1 <- ggplot(data = dt.full, mapping = aes(day, EnergyPurchase)) + geom_point() +
ggtitle("Average consumption for all houses during a year ") + xlab("Time")
+ ylab("Average consumption (kwh)")
+ geom_smooth(col=col.plot, se = T)

#AirTemperature EnergyPurchase
```

```
ggplot(data = dt.full, mapping = aes(AirTemperature, EnergyPurchase)) + geom_point()
+ ggtitle("AirTemperature against EnergyPurchased") + xlab("Outside airtemperature")
    + ylab("Average consumption per day (kwh)") + geom_smooth(col="green")

dt.id <- aggregate(cbind(EnergyLabelNumber, BBR_YearOfConstruction,
EnergyPurchaseScaled) ~ Id, data = watts.data, FUN = mean , na.rm =TRUE)

#Change from number to letter
dt.id$EnergyLabelNumber[which(dt.id$EnergyLabelNumber==1)] <- "A"
dt.id$EnergyLabelNumber[which(dt.id$EnergyLabelNumber==2)] <- "B"
dt.id$EnergyLabelNumber[which(dt.id$EnergyLabelNumber==3)] <- "C"
dt.id$EnergyLabelNumber[which(dt.id$EnergyLabelNumber==4)] <- "D"
dt.id$EnergyLabelNumber[which(dt.id$EnergyLabelNumber==5)] <- "E"
dt.id$EnergyLabelNumber[which(dt.id$EnergyLabelNumber==6)] <- "F"
dt.id$EnergyLabelNumber[which(dt.id$EnergyLabelNumber==7)] <- "G"

#Plot YOC
ggplot(dt.id, aes(BBR_YearOfConstruction)) + geom_histogram(bins = 15)
+ ggtitle("BBR") + xlab("Year of construction") + ylab("Frequency")

## PLOT WITH ENERGY LABEL ##
watts.data$EnergyLabel[watts.data$EnergyLabelNumber==1] <- "A"
watts.data$EnergyLabel[watts.data$EnergyLabelNumber==2] <- "B"
watts.data$EnergyLabel[watts.data$EnergyLabelNumber==3] <- "C"
watts.data$EnergyLabel[watts.data$EnergyLabelNumber==4] <- "D"
watts.data$EnergyLabel[watts.data$EnergyLabelNumber==5] <- "E"
watts.data$EnergyLabel[watts.data$EnergyLabelNumber==6] <- "F"
watts.data$EnergyLabel[watts.data$EnergyLabelNumber==7] <- "G"

watts.data$EnergyLabel <- as.factor(watts.data$EnergyLabel)

watts.plot1 <- aggregate(cbind(EnergyPurchaseScaled, AirTemperature, WindSpeed,
AtmosphericPressure,SurfaceSolarRadiationDownwards, IsWeekDay) ~ day, data =
subset(watts.data, EnergyLabel=="A"), FUN = mean , na.rm =TRUE)
watts.plot2 <- aggregate(cbind(EnergyPurchaseScaled, AirTemperature, WindSpeed,
AtmosphericPressure,SurfaceSolarRadiationDownwards, IsWeekDay) ~ day, data =
subset(watts.data, EnergyLabel=="B"), FUN = mean , na.rm =TRUE)
watts.plot3 <- aggregate(cbind(EnergyPurchaseScaled, AirTemperature, WindSpeed,
AtmosphericPressure,SurfaceSolarRadiationDownwards, IsWeekDay) ~ day, data =
subset(watts.data, EnergyLabel=="C"), FUN = mean , na.rm =TRUE)
watts.plot4 <- aggregate(cbind(EnergyPurchaseScaled, AirTemperature, WindSpeed,
AtmosphericPressure,SurfaceSolarRadiationDownwards, IsWeekDay) ~ day, data =
subset(watts.data, EnergyLabel=="D"), FUN = mean , na.rm =TRUE)
watts.plot5 <- aggregate(cbind(EnergyPurchaseScaled, AirTemperature, WindSpeed,
AtmosphericPressure,SurfaceSolarRadiationDownwards, IsWeekDay) ~ day, data =
subset(watts.data, EnergyLabel=="E"), FUN = mean , na.rm =TRUE)
watts.plot6 <- aggregate(cbind(EnergyPurchaseScaled, AirTemperature, WindSpeed,
AtmosphericPressure,SurfaceSolarRadiationDownwards, IsWeekDay) ~ day, data =
subset(watts.data, EnergyLabel=="F"), FUN = mean , na.rm =TRUE)
watts.plot7 <- aggregate(cbind(EnergyPurchaseScaled, AirTemperature, WindSpeed,
AtmosphericPressure,SurfaceSolarRadiationDownwards, IsWeekDay) ~ day, data =
subset(watts.data, EnergyLabel=="G"), FUN = mean , na.rm =TRUE)
```

```
#Individual slope
AuA <- coef(lm(EnergyPurchaseScaled ~ AirTemperature, data = watts.plot1))[[2]]
BuA <- coef(lm(EnergyPurchaseScaled ~ AirTemperature, data = watts.plot2))[[2]]
CuA <- coef(lm(EnergyPurchaseScaled ~ AirTemperature, data = watts.plot3))[[2]]
DuA <- coef(lm(EnergyPurchaseScaled ~ AirTemperature, data = watts.plot4))[[2]]
EuA <- coef(lm(EnergyPurchaseScaled ~ AirTemperature, data = watts.plot5))[[2]]
FuA <- coef(lm(EnergyPurchaseScaled ~ AirTemperature, data = watts.plot6))[[2]]
GuA <- coef(lm(EnergyPurchaseScaled ~ AirTemperature, data = watts.plot7))[[2]]

uA <- as.data.frame(rbind(AuA,BuA,CuA,DuA,EuA,FuA, GuA));
names(uA) = "uA coefficient"; uA

#EnergyPurchaseScaled Energylabel plot
par(mfrow=c(1,1))
boxplot(EnergyPurchaseScaled ~ EnergyLabel, data = watts.data, col = 2:7)

# Weather data ~ EnergyPurchaseScaled #
#AirTemperature
g1 <- ggplot(watts.plot1,aes(AirTemperature,EnergyPurchaseScaled,
colour = factor(IsWeekDay))) + geom_point() + ylim(0,1) + geom_smooth(se = FALSE,
method = "lm", color ="#99ff66") + ggtitle("EnergyLabel A")
+ ylab("Consumption") + theme(legend.position = "none")
+ scale_colour_manual(values=c("red","black"))

g2 <- ggplot(watts.plot2,aes(AirTemperature,EnergyPurchaseScaled,colour =
factor(IsWeekDay))) + geom_point() + ylim(0,1) + geom_smooth(se = FALSE,
method = "lm", color ="#99ff66") + ggtitle("EnergyLabel B")
+ ylab("Consumption") + theme(legend.position = "none")
+ scale_colour_manual(values=c("red","black"))

g3 <- ggplot(watts.plot3,aes(AirTemperature,EnergyPurchaseScaled,
colour = factor(IsWeekDay))) + geom_point() + ylim(0,1) + geom_smooth(se = FALSE,
method = "lm", color ="#99ff66") + ggtitle("EnergyLabel C")
+ ylab("Consumption") + theme(legend.position = "none")
+ scale_colour_manual(values=c("red","black"))

g4 <- ggplot(watts.plot4,aes(AirTemperature,EnergyPurchaseScaled,colour =
factor(IsWeekDay))) + geom_point() + ylim(0,1) + geom_smooth(se = FALSE,
method = "lm", color ="#99ff66") + ggtitle("EnergyLabel D")
+ ylab("Consumption") + theme(legend.position = "none")
+ scale_colour_manual(values=c("red","black"))

g5 <- ggplot(watts.plot5,aes(AirTemperature,EnergyPurchaseScaled,
colour = factor(IsWeekDay))) + geom_point() + ylim(0,1) + geom_smooth(se = FALSE,
method = "lm", color ="#99ff66") + ggtitle("EnergyLabel E") + ylab("Consumption")
+ theme(legend.position = "none") + scale_colour_manual(values=c("red","black"))

g6 <- ggplot(watts.plot6,aes(AirTemperature,EnergyPurchaseScaled,colour =
factor(IsWeekDay))) + geom_point() + ylim(0,1) + geom_smooth(se = FALSE, method =
"lm", color ="#99ff66") + ggtitle("EnergyLabel F") + ylab("Consumption") +
theme(legend.position = "none") + scale_colour_manual(values=c("red","black"))
```

```
g7 <-  ggplot(watts.plot7,aes(AirTemperature,EnergyPurchaseScaled,colour =
factor(IsWeekDay))) + geom_point() + ylim(0,1) + geom_smooth(se = FALSE, method =
"lm", color ="#99ff66") + ggtitle("EnergyLabel G") + ylab("Consumption") +
theme(legend.position = "none") + scale_colour_manual(values=c("red","black"))

grid.arrange(g1, g2, g3, g4, g5, g6, g7, ncol=2)
grid.text("Black indicating weekdays", x = unit(0.7, "npc"), y = unit(.18,
"npc"),gp = gpar(fontsize=14, fontfamily="Times New Roman"))
grid.text("Red indicating weekends", x = unit(0.695, "npc"), y = unit(.1,
"npc"),gp = gpar(fontsize=14, fontfamily="Times New Roman"))

uA <- as.data.frame(rbind(AuA,BuA,CuA,DuA,EuA,FuA, GuA));
names(uA) = "uA coefficient"; uA

s#Wind
g7 <- ggplot(watts.plot1,aes(WindSpeed,EnergyPurchaseScaled,colour =
factor(IsWeekDay))) + geom_point() + ylim(0,1) + geom_smooth(se = FALSE, method =
"lm", color ="blue") + ggtitle("EnergyLabel A") + ylab("Consumption")

g8 <- ggplot(watts.plot2,aes(WindSpeed,EnergyPurchaseScaled,colour =
factor(IsWeekDay))) + geom_point() + ylim(0,1) + geom_smooth(se = FALSE, method =
"lm", color ="blue") + ggtitle("EnergyLabel B") + ylab("Consumption")

g9 <- ggplot(watts.plot3,aes(WindSpeed,EnergyPurchaseScaled,colour =
factor(IsWeekDay))) + geom_point() + ylim(0,1) + geom_smooth(se = FALSE, method =
"lm", color ="blue") + ggtitle("EnergyLabel C") + ylab("Consumption")

g10 <- ggplot(watts.plot4,aes(WindSpeed,EnergyPurchaseScaled,colour =
factor(IsWeekDay))) + geom_point() + ylim(0,1) + geom_smooth(se = FALSE, method =
"lm", color ="blue") + ggtitle("EnergyLabel D") + ylab("Consumption")

g11 <- ggplot(watts.plot5,aes(WindSpeed,EnergyPurchaseScaled,colour =
factor(IsWeekDay))) + geom_point() + ylim(0,1) + geom_smooth(se = FALSE, method =
"lm", color ="blue") + ggtitle("EnergyLabel E") + ylab("Consumption")

g12 <- ggplot(watts.plot6,aes(WindSpeed,EnergyPurchaseScaled,colour =
factor(IsWeekDay))) + geom_point() + ylim(0,1) + geom_smooth(se = FALSE, method =
"lm", color ="blue") + ggtitle("EnergyLabel F") + ylab("Consumption")

g29 <- ggplot(watts.plot1,aes(WindSpeed,EnergyPurchaseScaled,colour =
factor(IsWeekDay))) + geom_point() + ylim(0,1) + geom_smooth(se = FALSE, method =
"lm", color ="blue") + ggtitle("EnergyLabel G") + ylab("Consumption")

grid.arrange(g7, g8, g9, g10, g11, g12, g29, ncol=2)

#Sun
g13 <- ggplot(watts.plot1,aes(SurfaceSolarRadiationDownwards,EnergyPurchaseScale
,colour = factor(IsWeekDay))) + geom_point() + ylim(0,1) + geom_smooth(se =
FALSE, method = "lm", color ="blue") + ggtitle("EnergyLabel A") + ylab("Consumption")

g14 <- ggplot(watts.plot2,aes(SurfaceSolarRadiationDownwards,EnergyPurchaseScale
```

```
,colour = factor(IsWeekDay))) + geom_point() + ylim(0,1) + geom_smooth(se =
FALSE, method = "lm", color ="blue") + ggtitle("EnergyLabel B") + ylab("Consumption")

g15 <- ggplot(watts.plot3,aes(SurfaceSolarRadiationDownwards,EnergyPurchaseScale
,colour = factor(IsWeekDay))) + geom_point() + ylim(0,1) + geom_smooth(se =
FALSE, method = "lm", color ="blue") + ggtitle("EnergyLabel C") + ylab("Consumption")

g16 <- ggplot(watts.plot4,aes(SurfaceSolarRadiationDownwards,EnergyPurchaseScale
,colour = factor(IsWeekDay))) + geom_point() + ylim(0,1) + geom_smooth(se =
FALSE, method = "lm", color ="blue") + ggtitle("EnergyLabel D") + ylab("Consumption")

g17 <- ggplot(watts.plot5,aes(SurfaceSolarRadiationDownwards,EnergyPurchaseScale
,colour = factor(IsWeekDay))) + geom_point() + ylim(0,1) + geom_smooth(se =
FALSE, method = "lm", color ="blue") + ggtitle("EnergyLabel E") + ylab("Consumption")

g18 <- ggplot(watts.plot6,aes(SurfaceSolarRadiationDownwards,EnergyPurchaseScale
,colour = factor(IsWeekDay))) + geom_point() + ylim(0,1) + geom_smooth(se =
FALSE, method = "lm", color ="blue") + ggtitle("EnergyLabel F") + ylab("Consumption")

g30 <- ggplot(watts.plot7,aes(SurfaceSolarRadiationDownwards,EnergyPurchaseScale
,colour = factor(IsWeekDay))) + geom_point() + ylim(0,1) + geom_smooth(se =
FALSE, method = "lm", color ="blue") + ggtitle("EnergyLabel G") + ylab("Consumption")

grid.arrange(g13, g14, g15, g16, g17, g18, g30, ncol=2)

#Histograms of energyconsumption
g19 <- ggplot(watts.data, aes(EnergyPurchase)) + geom_histogram(bins = 25) +
ggtitle("EnergyPurchase")

g20 <- ggplot(watts.data, aes(EnergyPurchaseScaled)) + geom_histogram(bins = 25)
+ ggtitle("EnergyPurchaseScaled")

grid.arrange(g19, g20 , ncol=2)

g21 <- ggplot(watts.data, aes(AirTemperature)) + geom_histogram(bins = 25)
    + ggtitle("WindSpeed")

g22 <- ggplot(watts.data, aes(WindSpeed)) + geom_histogram(bins = 25)
    + ggtitle("WindSpeed")

g23 <- ggplot(watts.data, aes(SurfaceSolarRadiationDownwards))
+ geom_histogram(bins = 25) + ggtitle("SurfaceSolarRadiationDownwards")

g24 <- ggplot(watts.data, aes(AtmosphericPressure)) + geom_histogram(bins = 25)
    + ggtitle("AtmosphericPressure")

g25 <- ggplot(watts.data, aes(WindDirectionDegree)) + geom_histogram(bins = 25)
    + ggtitle("WindDirectionDegree")

grid.arrange(g21, g22, g23, g24, g25, ncol=2)

#Aggregate BBR data
```

```
dt.bbr <- aggregate(cbind(Residents, SquareMeters, BBR_YearOfConstruction,
    EnergyLabelNumber) ~ Id, data = watts, FUN = mean , na.rm =TRUE)

#Plot different BBR data
bbr3 <- ggplot(dt.bbr, aes(BBR_YearOfConstruction)) + geom_histogram(bins = 25) +
  xlab("Year of construction") + ylab("Frequency") +
  scale_x_continuous(breaks=seq(1835,2020,25)) +
  scale_y_continuous(breaks = seq(0,14 ,2))

bbr2 <- ggplot(dt.bbr, aes(Residents)) + geom_histogram(bins = 10) +
  xlab("Residents") + ylab("Frequency")+
  scale_x_continuous(breaks=seq(0,10,1)) +
  scale_y_continuous(breaks = seq(0,26,4))

bbr1 <- ggplot(dt.bbr, aes(SquareMeters)) + geom_histogram(bins = 20) +
 xlab("SquareMeters") + ylab("Frequency") +
   scale_x_continuous(breaks=seq(0,300,50)) +
   scale_y_continuous(breaks = seq(0,12,2))

bbr0 <- ggplot(dt.bbr, aes(EnergyLabelNumber)) + geom_histogram(bins = 7) +
  xlab("EnergyLabelNumber") + ylab("Frequency") +
  scale_x_continuous(breaks=seq(0,7,1)) +
  scale_y_continuous(breaks = seq(0,25,4))
grid.arrange(bbr1, bbr2, bbr3, bbr0, ncol=2)
#####################################################

dynamic_days <- read_delim("dynamic_days2.txt",
     "\t", escape_double = FALSE, col_types = cols(Idnr = col_character(),
         slut = col_character(), start = col_character(),
         tal = col_skip()), trim_ws = TRUE)
dynamic_days <- dynamic_days[,1:3]
names(dynamic_days) <- c("HouseholdId","StartDate","EndDate")
#####
# New subset ####
uniq <- unique(dynamic_days$HouseholdId)
n.uniq <- length(uniq)
newdata = c()
for(i in 1:n.uniq){
dataset = subset(watts.data, watts.data$Id ==  uniq[i] &
    (watts.data$day <= dynamic_days$StartDate[i] |
    watts.data$day >= dynamic_days$EndDate[i]))
newdata = rbind(newdata,dataset)
}

#Create BBR tables ####
#Cleaned BBR data
dt.bbr <- aggregate(cbind(Residents, BBR_YearOfConstruction, EnergyLabelNumber) ~
    Id , data = newdata, FUN = mean , na.rm = TRUE)
#Uncleaned BBR data
dt.bbr1 <- aggregate(cbind(Residents, BBR_YearOfConstruction, EnergyLabelNumber) ~
    Id , data = watts.data, FUN = mean , na.rm = TRUE)
```

```
min.date = c(); max.date = c()
for (i in 1:n.uniq){
  data = subset(watts.data, watts.data$Id == uniq[i])
min.date[i] = min(data$day)
max.date[i] = max(data$day)
}
dt.bbr1$StartDate = as.character(as.Date(min.date))
dt.bbr1$EndDate = as.character(as.Date(max.date))

dt.bbr$EnergyLabelNumber =LETTERS[dt.bbr$EnergyLabelNumber]
table_bbr= cbind(dt.bbr[],dynamic_days[,2:3])
xtable(table_bbr, digits = 0)

dt.bbr1$EnergyLabelNumber =LETTERS[dt.bbr1$EnergyLabelNumber]
table_bbr1= cbind(dt.bbr1[])
xtable(table_bbr1, digits = 0)

#correlation #####
dt.cor = data.frame(watts.data$AirTemperature, watts.data$AtmosphericPressure,
    watts.data$WindSpeed, watts.data$TotalSkyDirectRadiationAtSurface,
    watts.data$SurfaceSolarRadiationDownwards, as.numeric(watts.data$Residents) )
cormatrix = cor(dt.cor)

xtable(cormatrix)
#####
# Scatterplots for transformations ####
plot1 = ggplot(dt, aes(x = AirTemperature, y= EnergyPurchaseScaled)) +
    geom_point() + geom_smooth(colour="olivedrab2") + ylab("Consumption")

plot2 = ggplot(dt, aes(x = WindSpeed, y= EnergyPurchaseScaled)) + geom_point()
    + geom_smooth(colour="olivedrab2")+ ylab("Consumption")

plot3 = ggplot(dt, aes(x = SurfaceSolarRadiationDownwards, y= EnergyPurchaseScaled))
    + geom_point() + geom_smooth(colour="olivedrab2")+ ylab("Consumption")

plot4 = ggplot(dt, aes(x = TotalSkyDirectRadiationAtSurface,
    y= EnergyPurchaseScaled))
    + geom_point() + geom_smooth(colour="olivedrab2")+ ylab("Consumption")

plot5 = ggplot(dt, aes(x = AtmosphericPressure, y= EnergyPurchaseScaled))
+ geom_point() + geom_smooth(colour="olivedrab2")+ ylab("Consumption")

grid.arrange(plot1,plot2,plot3,plot4,plot5, ncol=2)
```

A.2.6 Classification

```
#### Machine learning variable definitions and packages ####
source("MachinelearningPreamble.R") #machine learning defintions
#### Decission tree ####
```

```
classNames = c("1", "2", "3", "4", "5", "6", "7")
X_traindf$Residents <- as.numeric(X_traindf$Residents)
X_testdf$Residents <- as.numeric(X_testdf$Residents)

X_traindf <- cbind(y_train, X_traindf)
X_testdf <- cbind(y_test, X_testdf)

names(X_traindf) <- c("y_train","Id","AT","AP","WS","SS","EP","RE")
names(X_testdf) <- c("y_test","Id","AT","AP","WS","SS","EP","RE")
fm <- formula(classNames[y_train] ~ AT + WS + AP +
              SS + EP + RE)

#Fit decision tree
mytree <- rpart(fm, data=X_traindf, control=rpart.control(minsplit=400,
minbucket=50, cp=0.0001), parms=list(split='Gini'), method="class")

#Plot tree
prp(mytree, digits = 2, cex = 0.8)

#Predition with decission tree
y_test_est_dt <- as.factor(predict(mytree, newdata=X_testdf, type="vector"))

#Confusion matrix
CM(X_testdf, y_test, y_test_est_dt, "dt" , 0)

#### K-nearest neighbors ####
# Choose optimal neighbor
source("MachinelearningPreamble.R")
set.seed(123)
N <- length(watts.data3[,1])
M <- length(attributeNames[2:8])
C <- length(levels(y))
classLabels <- y;
classNames <- unique(classLabels);
y <- match(classLabels, classNames)
CV <- cvFolds(N, K=10);
K = 10

# K-nearest neighbors parameters
L = 50; # Maximum number of neighbors

# Variable for classification error
Error = array(rep(NA, times=K*L), dim=c(K,L))
a = array(rep(NA, times= K), dim=K)
b = array(rep(NA, times= K), dim=K)

#Crossvalidation to choose optimal number of neighbors
for(k in 1:K){ # For each crossvalidation fold

  print(paste('Crossvalidation fold ', k, '/', K, sep=''))

  # Extract training and test set
```

```
  X_train <- X[CV$which!=k, ];
  y_train <- y[CV$which!=k];
  X_test <- X[CV$which==k, ];
  y_test <- y[CV$which==k];
  CV$TrainSize[k] <- length(y_train)
  CV$TestSize[k] <- length(y_test)

  X_traindf <- data.frame(X_train[,2:7])
  X_testdf <- data.frame(X_test[,2:7])

  X_traindf <- as.data.frame(lapply(X_traindf,
        function(col) { col / max(abs(col)) }))
  X_testdf <- as.data.frame(lapply(X_testdf,
        function(col) { col / max(abs(col)) }))

  colnames(X_testdf) <- attributeNames[2:7]
  colnames(X_traindf) <- attributeNames[2:7]
  for(l in 1:L){ # For each number of neighbors

    # Use knnclassify to find the l nearest neighbors
    y_test_estt <- knn(X_traindf, X_testdf, cl=y_train, k = l, prob = FALSE,
        algorithm="kd_tree")

    # Compute number of classification errors
    Error[k,l] = sum(y_test!=y_test_estt); # Count the number of errors
  }
}
plot(colSums(Error)/sum(CV$TestSize)*100, main='Error rate', xlab='Number of
neighbors', ylab='Classification error rate (%)', pch=20, type='l');

K = 28 # Number of neighbors
# Use knnclassify to find the K nearest neighbors
#Normalize data

X_traindf <- data.frame(X_train[,2:7])
X_testdf <- data.frame(X_test[,2:7])

X_traindf <- as.data.frame(lapply(X_traindf, function(col) { col / max(abs(col)) }))
X_testdf <- as.data.frame(lapply(X_testdf, function(col) { col / max(abs(col)) }))

y_test_est_knn <- knn(X_traindf, X_testdf, cl=y_train, k = K, prob = FALSE,
    algorithm="kd_tree")

#Confusion Matrix KNN with K = 28
CM(X_test, y_test, y_test_est_knn, "knn")

#### Artificial neural network ####
#Function to find maximum probability in row
max_idx <- function(Y){
  # function to extract the column index of the max value for each row of Y
  #
  # Author: Laura Frølich, lff@imm.dtu.dk
```

```
  apply(Y, 1, which.max)
}

attributeNames <- attributeNames[-1]

X_train <- X_train[complete.cases(X_train),]
X_test <- X_test[complete.cases(X_test),]

# K-fold crossvalidation
K = 10;
set.seed(12345) # for reproducibility
CV <- cvFolds(N, K=K)
# set up vectors that will store sizes of training and test sizes
CV$TrainSize <- c()
CV$TestSize <- c()

# Parameters for neural network classifier
NHiddenUnits = 7;  # Number of hidden units
NTrain = 10; # Number of re-trains of neural network

# Variable for classification error
Error = rep(NA, times=K)
y_trainfact <- factor(y_train)


for(k in 1:K){ # For each crossvalidation fold
  print(paste('Crossvalidation fold ', k, '/', K, sep=''))

  # Extract training and test set
  X_train <- X[CV$which!=k, ];
  y_train <- y[CV$which!=k];
  X_test <- X[CV$which==k, ];
  y_test <- y[CV$which==k];
  CV$TrainSize[k] <- length(y_train)
  CV$TestSize[k] <- length(y_test)

  X_traindf <- data.frame(cbind(y_train,X_train[,2:7]))
  X_traindf$y_train <- as.factor(X_traindf$y_train)

  X_testdf <- data.frame(cbind(y_test,X_test[,2:7]))
  X_testdf$y_test <- as.factor(X_testdf$y_test)

  #Normalize data
  X_traindf <- cbind(X_traindf[,1], as.data.frame(lapply(X_traindf[,-1],
    function(col) { col / max(abs(col)) })))
  X_testdf <- cbind(X_testdf[,1], as.data.frame(lapply(X_testdf[,-1],
    function(col) { col / max(abs(col)) })))

  #Fit colnames
  colnames(X_traindf) <- c("y_trainfact",attributeNames)
  colnames(X_testdf) <- c("y_trainfact", attributeNames)
```

```
  # Fit neural network to training set
  MSEBest = Inf;
  for(t in 1:NTrain){  #rprop+

    (fmla <- as.formula(paste("y_trainfact ~ ", paste(attributeNames, collapse=
    "+"))))
    netwrk = nnet(fmla, X_traindf, size=NHiddenUnits,linear.output=F,
    err.fct='sse', maxit = 200, decay = 5e-5);
    Y_train_est <- predict(object=netwrk, newdata=X_traindf, type='class');
    y_train_est = as.numeric(Y_train_est)
    mse <- sum((y_train_est-as.integer(y_train))^2)

    if(mse<MSEBest){
      bestnet <- netwrk
      MSEBest <- mse
    }
  }

  # Predict model on test data

  computeres <- predict(object = bestnet, newdata = X_testdf[,-1], type =
  "class") #can also change to type = "raw" and then use max function of
  probabilities output
  y_test_est = as.integer(computeres)
  # Compute error rate
  Error[k] = sum((y_test-y_test_est)^2); # Count the number of errors

}

# Print the error rate
print(paste('Mean Sum of Squares Error (MSSE): ', round(sum(Error)
/sum(CV$TestSize),3), sep=''));

# Display the trained network (given for last cross-validation fold)
plot(bestnet)

#Calculate confusion matrix
CM(X_testdf, y_test, y_test_est, "dt", 0)

#Copying weights into dataframe
weights <- as.data.frame(matrix(data = NA, nrow = 15, ncol = 8))
weights[,1] <- c("[1]","[8]","[15]","[22]","[29]","[36]","[43]","[50]","[57]","[64]",
"[71]","[78]","[85]","[92]","[99]")
for ( i in 1:7){
  weights[1,i+1] <- bestnet$wts[i]
  weights[2,i+1] <- bestnet$wts[i+7]
  weights[3,i+1] <- bestnet$wts[i+14]
  weights[4,i+1] <- bestnet$wts[i+21]
  weights[5,i+1] <- bestnet$wts[i+28]
  weights[6,i+1] <- bestnet$wts[i+35]
  weights[7,i+1] <- bestnet$wts[i+42]
  weights[8,i+1] <- bestnet$wts[i+49]
```

```
  weights[9,i+1] <- bestnet$wts[i+56]
  weights[10,i+1] <- bestnet$wts[i+63]
  weights[11,i+1] <- bestnet$wts[i+70]
  weights[12,i+1] <- bestnet$wts[i+77]
  weights[13,i+1] <- bestnet$wts[i+84]
  weights[14,i+1] <- bestnet$wts[i+91]
  weights[15,i+1] <- bestnet$wts[i+98]
}

#Write to Latex output
xtable(weights)

#Simulate Sigmoid function
x <- seq(-10,10,by = 0.01)
sig <- 1/(1+exp(-x))
sig1 <- 1/(1+exp(-x + 3.5))
sig2 <- 1/(1+exp(-x - 3.5))

plot(x,sig, type = "l", xlab = "Input", ylab = "Output",
    main = "Sigmoid function with bias")
lines(x,sig1, col = "red")
lines(x,sig2, col = "blue")
```

A.2.7 Regression

```
#Simple linear regression ####
# Initialize variables
factors = c("AirTemperature")
full_coefs=0; sig = 0; r_sq=0; full_std = 0; dfs=0;res= 0; full_res= c();

par(mfrow=c(3,2)); par(mar=c(2,2,4,2))
#Create model for each house
for ( i in 1:n.uniq) {
  dataset = subset(newdata, newdata$Id == uniq[i])
  model_simple = lm(sqrt(EnergyPurchaseScaled) ~ AirTemperature, data = dataset)
 plot(model_simple, which=c(1:2), main = paste("HouseId: ", uniq[i]))
  coefs = model_simple$coefficients
  full_coefs = append(full_coefs, coefs )
  std_error <- summary(model_simple)[["coefficients"]][,2]
  full_std = append(full_std, std_error)
  res = c(1,as.vector(model_simple$residuals))
  full_res = c(full_res,res)
  sig[i] = summary(model_simple)$sigma
  r_sq[i] = summary(model_simple)$r.squared
  print( i)
  dfs[i] = model_simple$df.residual
}
#####
# Collect coefs, std and res####
```

```
start.ids <- which(names(full_coefs) == "(Intercept)")
end.ids <- c(which(names(full_coefs) == "(Intercept)")[-1] - 1, length(full_coefs))
mats <- mapply(function(x, y) t(full_coefs[seq(x, y)]), start.ids, end.ids)
m <- as.data.frame(t(rbind.fill.matrix(mats)))

start.ids <- which(names(full_std) == "(Intercept)")
end.ids <- c(which(names(full_std) == "(Intercept)")[-1] - 1, length(full_std))
matss <- mapply(function(x, y) t(full_std[seq(x, y)]), start.ids, end.ids)
ms <- as.data.frame(t(rbind.fill.matrix(matss)))

start.ids <- which(full_res == 1)
end.ids <- c(which(full_res == 1) -1, length(full_res)) [-1]
resid <- mapply(function(x, y) t(full_res[seq(x, y)]), start.ids, end.ids)
res <- as.data.frame(t(rbind.fill.matrix(resid)))
res = res[-1,]

xtable(cbind(m[1:18,], c(19:36), rbind(m[19:35,],c(0,0))), digits = 5)

r1 = r_sq
r1mean = mean(r_sq)
#####
#Plots ####
uA = as.numeric(m[,"V2"])
uA_std = as.numeric(ms[,"V1"])
uA_lwr = 0; uA_upr = 0
for(i in 1:n.uniq){
  uA_upr[i] = uA[i]+ qt(0.975,dfs[i])*uA_std[i]
  uA_lwr[i] = uA[i]- qt(0.975,dfs[i])*uA_std[i]
}
dt.simple = data.frame(uA, uA_std, uA_upr, uA_lwr)
 ggplot(dt.simple, aes(x=1:n.uniq, y=uA))+
  xlab("HouseId")+ylab(expression(paste("Estimates of ", beta, "uA")))+
  theme( legend.position = "none" ) +
  geom_segment(data = dt.simple, aes(x = 1:n.uniq, y = uA_lwr, xend = 1:n.uniq,
                                     yend = uA_upr), colour="red")+
  geom_hline(yintercept = 0, colour="olivedrab3", linetype = 2, size = 0.8) +
  scale_y_continuous(breaks = seq(-0.02,0.01 , 0.0025)) +
  scale_x_continuous(breaks=seq(1,44,2)) + geom_point()
#####
# Few plots of residuals  ####
 par(mfrow=c(3,2))
for ( i in c(7,26,35)) {
  dataset = subset(newdata, newdata$Id == uniq[i])
  model_simple = lm(sqrt(EnergyPurchaseScaled) ~ AirTemperature, data = dataset)
  plot(model_simple, which=c(1:2), main = paste("HouseId: ", uniq[i], "-", i))
}
#####
#Sign test####
sign = 0
for(i in 1:n.uniq){
  res_s = as.numeric(na.omit(res[,i]))
  n.res <- length(res_s)
```

```
  ### 95% interval:
  int = (n.res-1)/2 + 1.96 * sqrt((n.res-1)/4) * c(-1,1)
  ### test:
  num = sum(na.omit(res_s[-1] * res_s[-n.res] < 0  ))
  if(num <int[2] & num >int[1]){
    x = 1
  }  else {
    x = 0
  }
  sign[i]=x
}
sign
sum(sign)
#####
# Ks-test ####
set.seed(12345)
ks_result  = 0
for (i in 1:n.uniq){
  ks_test =ks.test(res[i],rnorm(700, 0, sig[i]))
  if(ks_test$p.value <= 0.05){
    ks = 0
  } else {
    ks = 1
  }
  ks_result[i] = ks
}
sum(as.numeric(format(round(ks_result, 3), nsmall = 2)))
##################################################

#Multiple linear regression ####
# Initialize variables #####
full_coefs=0; sig = 0; r_sq=0; full_std = 0; dfs=0;res= 0; full_res= c();
uniq = as.character(dynamic_days$HouseholdId); n.uniq = length(uniq)
factors = c("WindSpeed", "WindDirectionDescription", "SquareMeters",
"SurfaceSolarRadiationDownwards","AirTemperature",
"WindSpeed:AirTemperature", "AtmosphericPressure")

#Dynamic days ####
par(mfrow=c(3,2)); par(mar=c(2,2,4,2))
for ( i in 1:n.uniq) {
model_scale = model.select.linear(factors, watts.data,
    dynamic_days$StartDate[i], dynamic_days$EndDate[i] , uniq[i] ,2, 2, 1)
  if (i == 7 | i == 30 | i == 31){
plot(model_scale, which=c(1:2), main = paste("HouseId: ", uniq[i], "-", i))
  }
coefs = model_scale$coefficients
full_coefs = append(full_coefs, coefs )
std_error <- summary(model_scale)[["coefficients"]][,2]
full_std = append(full_std, std_error)
res = c(1,as.vector(model_scale$residuals))
full_res = c(full_res,res)
sig[i] = summary(model_scale)$sigma
```

```
r_sq[i] = summary(model_scale)$r.squared
print( i)
dfs[i] = model_scale$df.residual
}
# Collect coefs, std and res####
start.ids <- which(names(full_coefs) == "(Intercept)")
end.ids <- c(which(names(full_coefs) == "(Intercept)")[-1] - 1, length(full_coefs))
mats <- mapply(function(x, y) t(full_coefs[seq(x, y)]), start.ids, end.ids)
m <- as.data.frame(t(rbind.fill.matrix(mats)))

start.ids <- which(names(full_std) == "(Intercept)")
end.ids <- c(which(names(full_std) == "(Intercept)")[-1] - 1, length(full_std))
matss <- mapply(function(x, y) t(full_std[seq(x, y)]), start.ids, end.ids)
ms <- as.data.frame(t(rbind.fill.matrix(matss)))

start.ids <- which(full_res == 1)
end.ids <- c(which(full_res == 1) -1, length(full_res)) [-1]
resid <- mapply(function(x, y) t(full_res[seq(x, y)]), start.ids, end.ids)

#get residuals
res <- as.data.frame(t(rbind.fill.matrix(resid)))
res = res[-1,]

#Get standard deviation
std = t(ms[2,])

#Get R-squared
r2 = r_sq
(r2-r1)
mean(r_sq)


# create dataframe for plots#####
uA = as.numeric(m["AirTemperature",])
uA_std = as.numeric(ms["AirTemperature",])
ws = as.numeric(m["WindSpeed",])
ws_std = as.numeric(ms["WindSpeed",])
sol = as.numeric(m["SurfaceSolarRadiationDownwards",])
sol_std = as.numeric(ms["SurfaceSolarRadiationDownwards",])
uAws = as.numeric(m["AirTemperature:WindSpeed",])
uAws_std = as.numeric(ms["AirTemperature:WindSpeed",])
atmos = as.numeric(m["AtmosphericPressure",])
intercept = as.numeric(m["(Intercept)",])

uA_upr=0;uA_lwr=0;sol_upr=0;sol_lwr=0;ws_upr=0;ws_lwr=0; uAws_upr=0 ; uAws_lwr =0
for(i in 1:n.uniq){
  sol_upr[i] = sol[i]+ qt(0.975,dfs[i])*sol_std[i]
  sol_lwr[i] = sol[i]- qt(0.975,dfs[i])*sol_std[i]
  ws_upr[i] = ws[i]+ qt(0.975,dfs[i])*ws_std[i]
  ws_lwr[i] = ws[i]- qt(0.975,dfs[i])*ws_std[i]
  uA_upr[i] = uA[i]+ qt(0.975,dfs[i])*uA_std[i]
  uA_lwr[i] = uA[i]- qt(0.975,dfs[i])*uA_std[i]
```

```
  uAws_upr[i] = uAws[i]+ qt(0.975,dfs[i])*uAws_std[i]
  uAws_lwr[i] = uAws[i]+ qt(0.975,dfs[i])*uAws_std[i]
}
plotdata = data.frame(uA,uA_std,ws,ws_std,sol, sol_std, uniq,sol_upr,sol_lwr,ws_upr,
        ws_lwr,uA_lwr, uA_upr, uAws,uAws_std)

#Plots of estimates ####
est_sol = ggplot(plotdata, aes(x=1:n.uniq, y=sol))+
  xlab("HouseId")+ ylab(expression(paste("Estimates of ", beta, "gA")))
    + theme( legend.position = "none")
    + geom_segment(data = plotdata, aes(x = 1:n.uniq, y = sol_upr, xend = 1:n.uniq,
    yend = sol_lwr), colour="red")
    + geom_hline(yintercept = 0, colour="olivedrab3", linetype = 2, size = 0.8)
    + scale_x_continuous(breaks=seq(1,44,2)) + geom_point()

est_uA = ggplot(plotdata, aes(x=1:n.uniq, y=uA))
    + xlab("HouseId")+ ylab(expression(paste("Estimates of ", beta, "uA")))
    + theme( legend.position = "none")
    + geom_segment(data = plotdata, aes(x = 1:n.uniq, y = uA_upr, xend = 1:n.uniq,
    yend = uA_lwr), colour="red")
    + geom_hline(yintercept = 0, colour="olivedrab3", linetype = 2, size = 0.8)
    + scale_x_continuous(breaks=seq(1,44,2)) +  geom_point()

est_ws = ggplot(plotdata, aes(x=1:n.uniq, y=ws))
    + xlab("HouseId")+ylab(expression(paste("Estimates of ", beta, "ws")))
    + theme( legend.position = "none")
    + geom_segment(data = plotdata, aes(x = 1:n.uniq, y = ws_upr, xend = 1:n.uniq,
 yend = ws_lwr), colour="red" )
    + geom_hline(yintercept = 0, colour="olivedrab3", linetype = 2, size = 0.8)
    + scale_x_continuous(breaks=seq(1,44,2)) + geom_point()

#Sign test####
sign1 = 0
for(i in 1:n.uniq){
res_s = as.numeric(na.omit(res[,i]))
n.res <- length(res_s)
### 95% interval:
int = (n.res-1)/2 + 1.96 * sqrt((n.res-1)/4) * c(-1,1)
### test:
num = sum(na.omit(res_s[-1] * res_s[-n.res] < 0  ))
if(num <int[2] & num >int[1]){
  x = 1
}  else {
  x = 0
}
sign1[i]=x
}
sign1
sum(sign1)
#####
# Ks-test ####
set.seed(12345) # For reproduction results in rnorm
```

```
ks_result1  = 0
for (i in 1:n.uniq){
ks_test1 =ks.test(res[i],rnorm(1000, 0, sig[i]))
if(ks_test1$p.value <= 0.05){
  ks = 0
} else {
  ks = 1
}
ks_result1[i] = ks
}
sum(as.numeric(format(round(ks_result1, 3), nsmall = 2)))
```

A.2.8 Clustering

```
source("setup.R")
source("LoadData.R")
dt.clust <- aggregate(cbind(Residents, BBR_YearOfConstruction, EnergyLabelNumber,
            EnergyPurchaseScaled) ~ Id , data = newdata, FUN = mean , na.rm = TRUE)
dt.clust1 <- aggregate(cbind(EnergyLabelNumber, EnergyPurchaseScaled) ~ Id,
data = newdata, FUN = mean , na.rm = TRUE)
class = dt.clust[,4]

sol[is.na(sol)] <- 0
ws[is.na(ws)] <-0
atmos[is.na(atmos)]<- 0
uAws[is.na(uAws)]<-0

#With BBR
Xdf = data.frame(dt.clust[,-c(1,4)], uA, sol, ws, atmos)
#Scale
Xdf = cbind(Xdf, as.data.frame(lapply(Xdf[,-1], function(col){col/max(abs(col))})))


#Without BBR
Xdf = data.frame(dt.clust1[,c(3)], uA, sol, ws, atmos, intercept)
#Scale
Xdf = as.data.frame(lapply(Xdf, function(col){col/max(abs(col))}))

# number of clusters
K=7

#GMM
set.seed(12554)
model <- Mclust(data=Xdf, G=K) # using the mclust package

# Get clustering
i = as.vector(model$classification) # using the mclust package
which(i==class)
# Get cluster centers
```

```
Xc = t(model$parameters$mean) # using the mclust package

## Plot results
Xc= as.data.frame(Xc)
levels(Xc) = c(4,5,1,6,2,7,3)
TrueClasses = as.factor(class)
Clusters = as.factor(i)

GMMplot=
  ggplot(data = Xdf, aes(x=uA, y=EnergyPurchaseScaled)) +
  geom_point(aes(color=Clusters), size=3.8) +
  geom_point(aes(color=TrueClasses), size=2) +
  geom_point(data = Xc, aes(x=uA, y=EnergyPurchaseScaled, color=as.factor(1:7)),
  shape=8, size=4.5) + ggtitle("Gaussian Mixture Models")+
  guides(fill=guide_legend(title="New Legend Title"))


#K-means ####
set.seed(123)
kmean = kmeans(Xdf, 7, nstart=2000)
# get cluster means
aggregate(Xdf,by=list(kmean$cluster),FUN=mean)
# append cluster assignment
mydata <- data.frame(Xdf, kmean$cluster)
clusplot(Xdf, kmean$cluster, color=TRUE, shade=TRUE,
         labels=2, lines=0, ylim=c(-3,4), xlim=c(-3,4))

# Plot results
clusterplot(data.frame(Xdf), class, kmean$cluster, kmean$centers, main='K-means');
Clusters = as.factor(kmean$cluster)
centers= as.data.frame(kmean$centers)
plotcluster(Xdf,kmean$cluster)
kmeanplot = ggplot(data = Xdf, aes(x=uA, y=EnergyPurchaseScaled)) +
  geom_point(aes(color=Clusters), size=3.8) +
  geom_point(aes(color=TrueClasses), size=2.1) +
  geom_point(data=centers, aes(x=uA, y=EnergyPurchaseScaled, color=as.factor(1:7)),
  shape=8, size=4.5) +
  ggtitle("K-means")# +theme( legend.position="none")
```

A.2.9 Testing linear regression with boxcox

```
######Box-cox transformed ####
dataset = subset(watts.data, watts.data$Id ==  uniq[2]&
!(watts.data$day <= dynamic_days$start[2] & watts.data$day >= dynamic_days$slut[2]))
lam = 1
for(i in 1:n.uniq){
 dataset =  subset(watts.data, watts.data$Id == uniq[i] &
 !(watts.data$day <= dynamic_days$StartDate[i] &
 watts.data$day >= dynamic_days$EndDate[i]))
```

```
 mod = lm(EnergyPurchase ~1 ,data=dataset)
 modbox = boxcox(mod)
 lambda = modbox$x[which(modbox$y == max(modbox$y))];
 lam[i] = lambda
}
##### box-cox function #####
model.select.linear.box= function(factors, data, starttime, endtime, Idnum,
scale_unscaled, criteria, lambda){
  dataset = as.data.frame(subset(data, data$Id == Idnum &
  (data$day <= starttime | data$day >= endtime)))
  attach(dataset)
  EnergyPurchase_scaled = EnergyPurchase/SquareMeters
  if (scale_unscaled == 1){  #unScaled
       if(lambda >= -0.1 & lambda <= 0.1){
        response = log(EnergyPurchase)
        print("log")
        } else{
        if(lambda >= 0.9 & lambda <= 1.1){
          response = EnergyPurchase
          print("none")
        } else{
          response = (EnergyPurchase^lambda - 1)/lambda
          print("k")
        }
        }
    model = lm(response ~ AirTemperature, data=dataset)
    scopes =  as.formula(paste("~", paste(factors, collapse="+")))

    } else {                    #scaled

     if(lambda >= -0.1 & lambda <= 0.1){
      response = log(EnergyPurchase_scaled)
    } else{
      if(lambda >= 0.9 & lambda <= 1.1){
        response = EnergyPurchase_scaled
      } else{
        response = (EnergyPurchase_scaled^lambda - 1)/lambda
      }
    }
    model = lm(response ~ AirTemperature, data=dataset)
    scopes =  as.formula(paste("~", paste(factors, collapse="+")))
  }


  while(criteria==1){
    x = add1(model, scopes, test="F")
    F_value = min(x[,6], na.rm=T)
    var = rownames(x[which.min(x[,6]),])
    if(F_value >= 0.051)
      break
    #   else if ( aic >= x[1,5])
    #      break
```

```
    formul <- as.formula(paste(".~.+",var))
    model = update(model,formul)
  }


  while(criteria==2){
    x = add1(model,scopes)
    aic = min(x[,4])
    var = rownames(x[which(x[,4]==min(x[,4])),])
    if(var == "<none>")
      break
    else if ( aic >= x[1,4])
      break
    formul <- as.formula(paste(".~.+",var))
    model = update(model,formul)
  }

  detach(dataset)
  return(model)
}

for ( i in 1:n.uniq){
  model_scale = model.select.linear.box(factors, watts.data, dynamic_days$EndDate[i],
  dynamic_days$EndDate[i] , uniq[i] , 1,  1, lam[i])
  coefs = model_scale$coefficients
  full_coefs = append(full_coefs, coefs )
  sig[i] = summary(model_scale)$sigma
  r_sq[i] = summary(model_scale)$r.squared
  print( i)
}

start.ids <- which(names(full_coefs) == "(Intercept)")
end.ids <- c(which(names(full_coefs) == "(Intercept)")[-1] - 1, length(full_coefs))
mats <- mapply(function(x, y) t(full_coefs[seq(x, y)]), start.ids, end.ids)
m <- t(rbind.fill.matrix(mats))

mean(r_sq)
median(r_sq)
```

Bibliography

Books

[8] Alan Agresti. *An introduction to categorical data analysis*. Wiley, 2007. ISBN: 0471226185.

[14] Nikhil Buduma. *Fundamentals of Deep Learning*. Taylor & Francis e-Library, 2004. ISBN: 0203451511.

[15] Michael J. Crawley. *Statistics: An Introduction using R*. Wiley, 2005. ISBN: 0470022973.

[17] Kevin Gurney. *An intro neural networks*. Taylor & Francis e-Library, 2004. ISBN: 0203451511.

[18] F.E. Harrell. *Regression Modeling Strategies :* eng. Springer, 2001, 568 s. ISBN: 0387952322.

[19] Jeff Heaton. *Introduction to Neural Networks for Java, 2nd Edition*. Pearsons Education Limited, 2004. ISBN: 1604390085.

[20] David W. Hosmer. *Applied Logistic Regression*. Wiley, 2013. ISBN: 9780470582473.

[21] Andrea Isoni. *Machine learning for the web*. Packt Publishig Ltd, 2016. ISBN: 1785886607.

[23] Phillip K. Janert. *Data Analysis with Open Source Tools*. O'Reilly Media Inc., 2010. ISBN: 0596802356.

[24] Henrik Madsen. *Time series analysis*. eng. Chapman Hall/CRC, 2008. ISBN: 142005967.

[26] Simon J. Sheather. *A Modern Approach to Regression with R*. Springer, 2009. ISBN: 0387096070.

[27] Tan Steinbach Kumar. *Introduction to Data Mining*. Pearsons Education Limited, 2004. ISBN: 1292026152.

[30] Alice Zheng. *Evaluating Machine Learning Methods*. O'Reilly Media Inc., 2015. ISBN: 1491932469.

Other

[1] URL: `http://4.bp.blogspot.com/-zuCQBrN8990/VGt45PZHXhI/AAAAAAAAA1E/jtQQaAj-PMc/s1600/gmm2.png` (visited on June 13, 2017).

[2] URL: http://courses.ee.sun.ac.za/Pattern_Recognition_813/lectures/lecture06/img2.png (visited on June 13, 2017).

[3] URL: https://docs.wso2.com/display/ML100/Model+Evaluation+Measures (visited on June 13, 2017).

[4] URL: https://www.ted.com/talks/kenneth_cukier_big_data_is_better_data?language=da (visited on June 13, 2017).

[5] URL: http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html (visited on June 20, 2017).

[6] URL: http://ai.stanford.edu/~chuongdo/papers/em_tutorial.pdf (visited on June 20, 2017).

[7] Lecture notes 02450 - Introduction to Machine Learning and Data Mining. *Decision trees and linear regression.*

[10] Stig Uffe Pedersen / Henrik Andersen. *Bekendtgørelse om Håndbog for Energikonsulenter (HB2016).* URL: https://www.retsinformation.dk/Forms/R0710.aspx?id=176520#id6b1cc506-4a65-41a0-9cb6-992f84d7d43a (visited on March 27, 2017).

[12] Boliga.dk. *Grafer over fordelling af byggeår.* URL: http://www.boliga.dk/graph.aspx (visited on June 5, 2017).

[13] Boliga.dk. *Hvad er Energimærkning?* URL: http://www.boliga.dk/energimaerke.aspx (visited on June 5, 2017).

[16] Energistyrelsen. *Energimærkning - værd at vide, når du sælger din bolig.* URL: http://sparenergi.dk/sites/forbruger.dk/files/contents/publication/energimaerkning-naar-du-saelger-bolig/energimaerkning-naar-du-saelger-bolig.pdf (visited on March 27, 2017).

R-packages

[9] Andreas Alfons. "cvTools: Cross-validation tools for regression models". R package version 0.3.2. 2012. URL: https://CRAN.R-project.org/package=cvTools.

[11] Alina Beygelzimer et al. "FNN: Fast Nearest Neighbor Search Algorithms and Applications". R package version 1.1. 2013. URL: http://CRAN.R-project.org/package=FNN.

[25] R Core Team. "R: A Language and Environment for Statistical Computing". Vienna, Austria, 2015. URL: https://www.R-project.org/.

[28] Terry Therneau, Beth Atkinson, and Brian Ripley. "rpart: Recursive Partitioning and Regression Trees". R package version 4.1-10. 2015. URL: https://CRAN.R-project.org/package=rpart.

[29] W. N. Venables and B. D. Ripley. "Modern Applied Statistics with S". ISBN 0-387-95457-0. New York, 2002. URL: http://www.stats.ox.ac.uk/pub/MASS4.