****

Short-term electric energy load forecasting for commercial buildings

David Sal

**3rd Year Project Interim Report**

Department of Electronic &  
Electrical Engineering

UCL

Supervisor: Dr Ryan Grammenos

30 March 2022

I have read and understood UCL’s and the Department’s statements and guidelines concerning plagiarism.

I declare that all material described in this report is my own work except where explicitly and individually indicated in the text. This includes ideas described in the text, figures and computer programs.

This report contains 27 pages (excluding this page and the appendices) and 9265 words.

Signed: David Sal Date: January 2022

|  |  |
| --- | --- |
|  |  |

Short term electric energy load forecasting for commercial buildings

David Sal

# Abstract

**Forecasting energy consumption accurately and applying proper energy management strategies allow for decreased energy usage leading to decreased environmental impact of the building and decreased operating costs. This work aims to develop a machine learning model that accurately predicts electric energy load 24 hours ahead. The dataset used for training and validation consists of 2 years of electricity load measurements taken every half an hour from two building blocks at an industrial site of IBM located in the UK. To increase forecasting accuracy the dataset has been enriched with exogenous factors such as weather conditions and holidays. This work consists of 2 stages starting with pre-processing the dataset. In the second stage multiple machine learning algorithms (Random Forest, Support Vector Machine, K Nearest Neighbour and Artificial Neural Networks) were developed. For the low-tech building block the Random Forest model performed the best with a mean absolute percentage error (MAPE) of 5.3%. For the modern building block Support Vector Machine model proved to be the most accurate with a MAPE of 2.87%. The two models have been analysed using SHAP interpretability technique to gain a deeper understanding of their inner workings.**

# Introduction

Energy has always been one of the most precious resources on earth and in the past few decades its demand has seen an exponential increase over time. According to [1] 40% of total energy use in Europe is made up by buildings whilst emitting 36% of the total emissions. According to [2] buildings use 20% more energy than required due to human errors, malfunctioning equipment, and faulty control systems. This surge in energy demand will continue in the future due to growing population and spread of energy demanding technology [3, 4, 5], thus using energy efficiently has caught the attention of many researchers. Forecasting energy consumption accurately and combining it with proper energy management strategies can benefit energy efficiency significantly. Accurate forecasts and proper energy management strategies would allow for decreased energy usage thus reducing the environmental effect of buildings and presenting economic benefits due to decreased operating costs [6].

There are two main approaches of forecasting the energy demand of a building: deterministic and data driven techniques. Deterministic techniques rely on simulations as demonstrated in [7] where the building characteristics are modelled using building physics. These are transparent and require no training data. Furthermore, it offers generalization as it can be easily applied to different buildings. The main disadvantage of this method is that it is very challenging to model the real world as the energy demand is influenced by numerous factors such as weather conditions, the number of occupants in the building, the thermal properties of the building and many others. These make simulations extremely difficult and introduce noteworthy inaccuracies into the forecasts [5].

Data driven techniques rely on historical data and make predictions using statistical or machine learning methods. These techniques are computationally less expensive and faster once the model is trained when compared to deterministic methods whilst generally being more accurate. With the advances of sensing technology most buildings are equipped with sensors that measure and store energy data. According to [8] these sensors generate multiple Zettabytes worth of data worldwide, which can be analysed to gain insightful information that could improve the energy efficiency of the building. Despite having large quantities of historical data available making accurate predictions remains challenging due to being influenced by multiple exogenous factors. This work aims to tackle this challenge using complex machine learning algorithms and artificial neural networks to develop a data driven model to make 1 day ahead forecasts.

## Problem statement and objectives

This project is based on a dataset provided by the industrial partner IBM and is made up of energy load measurements from two building blocks located on the Hursley industrial site in the UK. Measurements are taken every 30 mins over a period of 2 years from 1st Jan 2018 to 31st Dec. 2019. The first building block is named Hursley House (HH). It is low tech in its operation as the heating is via a wet system and ventilation is predominantly natural. This leads to the energy consumption being more dependent on the behaviour of the occupants that introduces more randomness into the electricity dataset. The second building block is named D East block (DE) and is more modern in its operation. The ventilation is mechanical with an HVAC plant. It is important to note that the first 5 months of data is missing for the DE block.

Given the raw nature of the massive data captured by sensors, unprocessed datasets tend to have a great number of inconsistent values thus, the project will need to start off with an extensive data pre processing stage. We will need to explore the dataset using visualisation and statistical techniques to identify missing values and outliers. The dataset needs to be cleaned from these inconsistent datapoints. Following this we move on to the second stage of the project and use the clean data to train a machine learning model that makes 1 day ahead forecasts. The main objectives for the projects are the following:

1. Identifying and handling inconsistent datapoints in the raw sensory data using Exploratory Data Analysis (EDA). When modifying data, it should be justified, and the modification should aim to restore original data.
2. Enriching the dataset with data about exogeneous factors such as air humidity and outside temperature. The exogeneous dataset should have a sufficiently small time interval between measurements to show changes throughout the day.
3. Exploring various machine learning models used for forecasting energy load. The models should be compared and the best one should be selected to make forecasts on unseen data. To maximize accuracy, feature engineering and hyperparameter finetuning should be performed. The accuracy, measured in mean absolute percentage error (MAPE), should optimally be less than 5%.
4. Giving an in-depth analysis of the final models developed for the two building blocks using interpretability techniques. The analysis should investigate the relationship between the target variable and specific features and explain how decisions are made locally and globally.

## Contributions

This work is a continuation of the MSc project of Kalliga, Polyxeni [9]. In her work an automated algorithm was proposed for cleaning the dataset that was replicated according to the descriptions of a flow chart on page 31 in [9]. The reproduced result did not match with the original one thus, whilst reusing the main ideas of the cleaning algorithm some parts needed modifications. After arranging a meeting to discuss the differences it was identified that the reason for the mismatch was due to incorrect descriptions on the flowchart, but our cleaned datasets were identical except for two instances that will be described later in the report.

As the focus of this work was different from the one in [9] everything done after the pre-processing stage is original work with guidance and suggestions from my supervisor, Dr Ryan Grammenos.

# Literature review and background theory

In this section we will explore what has been done in research area by giving an extensive literature review of the most relevant works. Furthermore, we introduce all the necessary background theory needed to understand this report.

## Literature review

As mentioned above, energy load forecasting has been of great importance and gained lot of attention in recent years. Many forecasting techniques emerged in the past decades that can accurately predict energy load based on historical data. In this work we will focus on time series forecasting methods given our time series data. We can divide the analysis of time series data into 2 important steps: First we need to obtain the structure of the underlying pattern of the given dataset. This can be done by decomposing our time series data into three components: trend, seasonality, and residual. The trend is the general movement of the dependant variable. The seasonality is the periodic fluctuation of the variable whilst the residual accounts for the remaining unexplainable parts of the variable. The more complex part of creating a forecasting model is the second step where we try to fit a model to make predictions for the future. We will discuss the most popular forecasting methods by presenting case studies for each. A more detailed description for each method can be found in the “Background theory” section.

One of the most popular forecasting methods is using Artificial Neural Networks (ANN). Nizami and Al Garni give an extensive explanation of ANN in their paper [10]. In their work they presented a two layered feedforward ANN to predict electrical energy consumption. For making predictions they used multiple exogenous factors as inputs to the NN. The results were compared to a regression model where the ANN significantly outperformed the regression model. Model adequacy is tested using visual inspection and the chi square test whilst for model validation predicted values are compared to values outside the training dataset. However, there is no further investigation of how inputs and outputs are related making the algorithm behave entirely like a black box. In a different case study conducted by Karatasou et al. [11] further improves the accuracy of ANN method by combining it with statistical methods such as hypothesis tests and cross validation as guidance for model selection. Two datasets are used for this research both consisting of hourly measurements for a period of over a year. The inputs to the NN are made up of exogenous factors, timestamp, and values from the previous timestamps to predict hourly load profiles using a feedforward neural network for modelling. They propose and analyse multiple models and achieve MAPE scores ranging from 2.5-16%. This paper puts lots of effort into validation of the model and gives great attention to input selection by finding the relationship between inputs and output. However, data pre-processing was neglected making the model prone to fail on datasets with significant outliers.

Support Vector Machines (SVM) are a further model frequently used for energy load forecasting. Xuemei et al. give an extensive overview of the mathematics behind this model in [12]. In their work they propose a Least Square SVM model to forecast cooling load of an office building using exogenous factors. Their timespan of their dataset was 5 months and they managed to achieve an impressive 1.65% MAPE score. This was possible as the dataset had low amount of seasonality and insignificant residual. Whilst they successfully show that the results are superior in comparison to a back propagation ANN their work does not investigate the relationship between input and output values. A similar case study presented by Fu et al. [13] SVM was used to forecast energy demand 24 hours ahead for different building subsystems. The dataset used was over a timeframe of 1 year with hourly measurements. This work compares SVM to ANN and two further methods (ARIMAX and REPTree). Whilst the paper successfully demonstrates the superiority of SVM related to the other methods it lacks the same analysis as in [12].

Another popular machine learning model is the K-Nearest Neighbour (KNN). Lora et al. [4] proposes a KNN model to make day ahead forecasts for electric energy data for Spain. The dataset spans over a timeframe of 2 years and has insignificant residual characteristics. The results are compared to a dynamic regression model, and it has been shown that KNN outperforms it with a MAPE score of 2.3%. As studies described above, this work does not interpret the developed model, nor does it compare it with multiple possible models except for dynamic regression.

A further favoured machine learning model used for time series forecasting is Random Forest (RF). Wang et al. [14] gives a great overview of the Random Forest method in their paper. They proposed an RF model for hourly building energy load forecasts and compared it to SVM. The dataset used had a high level of residual characteristics and span over a timeframe of 1 year. The results show that the RF model with a MAPE score of 7.75% outperform the SVM model. The paper provides a great insight into the feature importance of the proposed model but shows poor data pre-processing style as outliers are simply removed from the dataset without any attention to the time series nature of the dataset.

When conducting literature review, we could see multiple machine learning models being successfully applied to make satisfactory predictions on building energy data. Multiple studies tried to convince the reader of the superiority of a specific method whilst contradicting one another. From this we can conclude that the performances of machine learning models are dependant on the case study and the characteristics of the dataset thus, multiple forecasting methods need to be examined before making a choice. Furthermore, we could see big differences in performance with MAPE scores ranging from less than 1% to as high as ones in the 20% range. When inspecting the datasets, we could infer that the performance was primarily determined by the level of seasonality and residual in the time series data.

Overall, we could see that a lot of work has been conducted around= short-term building energy forecasting but most of them showed room for improvement. The most common weaknesses were the following:

* Missing the data pre-processing stage or poor performance when dealing with outliers (just removing them) and missing datapoints (imputing dataset mean)
* Being limited by one machine learning model and not exploring other methods
* Dealing with black box models and not providing an in-depth analysis using interpretability techniques

Inspired by the points above this work aim to fill the gap by presenting a sophisticated data pre-processing stage, exploring multiple machine learning models and giving an in-depth review of the proposed models.

## Background theory

The following section gives an overview of all the relevant background theory required for this project.

### Data pre-processing

Data pre-processing is the most important step of any data analytics process. Before performing any actions on a dataset, one needs to explore it identify patterns and clean the data from any inconsistent values. The steps of data pre-processing are outlined in the following sections.

1. Exploratory Data Analysis (EDA)

The first step of data pre-processing is to explore the data by visualising points in the dataset. Depending on the nature of the data this can be done by using statistical visualizations such as box plots, scatterplots, pie charts or histograms. This step helps us to familiarize ourselves with the data, identify hidden patterns, detect outliers, and make assumptions about the dataset.

1. Identifying inconsistent datapoints

Inconsistent points are outliers and missing points in the dataset. Outliers are unrealistic values that can be caused by human errors, malfunctioning equipment, faulty sensors, or control system with incorrect configurations. They can be detected by visualisation or by probabilistic methods that determine how likely the existence of the specific datapoint is. After choosing a probabilistic method for likeliness calculation and defining a threshold for what probability shall be considered as an outlier they can be easily identified. The most common probabilistic method is calculating the Z-score which can be obtained by subtracting the mean of the dataset and dividing by the standard deviation as shown in equation (1):

(1)

The larger the absolute value of the Z-score the more likely it is that the observation is an outlier. The threshold is usually set to be 3. Given the sensitivity of the mean to outliers this method can be easily skewed thus the mean can be replaced by the median or the mean of only the numbers inside the interquartile range which brings us to the next probabilistic method.

The interquartile range (IQR) can be obtained by placing all observations in ascending order and subtracting the value at 25% of the data length from the one at 75%. The value at 25% is denoted as Q1 whilst the one at 75% is denoted as Q3.

The outlier fences can be defined by the following equations (2,3,4):

(2)

(3)

(4)

Where *c* is a constant and usually chosen to be 1.5.

1. Cleaning the dataset

Cleaning the dataset of inconsistent points is of great significance before performing any further action on it. Understanding the reason behind an outlier is of great importance since they might contain crucial information about the building management system. Deciding what to do with an outlier can hugely affect the results of the following steps of data analysis. Removing outliers can lead to information getting lost whilst keeping an outlier can distort statistical analysis on the dataset due to most of those techniques being sensitive to outliers. Furthermore, in the case of time series data outliers cannot be simply removed as that would result in a missing point. Missing points for time series data is a further issue as we need to decide what value to impute when filling them. Common methods are to fill them using the mean of the dataset or based on adjacent values. When imputing artificial values, the challenge is to not distort the pattern of neighbouring values and not to change statistical features of the dataset either.

1. Data enrichment

Data enrichment is a process where we add additional information to our data that might be an influential input to the following machine learning (ML) models or Neural Networks (NN). For energy forecasting this information might be weather conditions (temperature, humidity, solar radiation, windspeed), occupancy, week of the day, month, season, holidays etc. Using these, the models can be improved which result in more accurate predictions as proven by many works mentioned in the literature review as shown for example in [10, 14, 15].

1. Transforming the dataset

Before moving on to creating an energy consumption model the dataset needs to be transformed into a form that can be fed into a ML algorithm or used for training for a NN. This requires the dataset to be in one data frame with values of similar nature having the same units and we need to normalize the values for all features.

### Time series forecasting methods

Artificial Neural Network – ANN

ANNs are modelling techniques that are trying to mimic the human brain. Just as a biological brain it is made up of large amount of processing units namely, neurons. These neurons are connected in parallel and process information using the weights attached to each neuron. These weights can be obtained by feeding historical data into the model whilst adjusting the weights so that the predicted output gets closer to the desired output. This process is called training of the network and is done by trying to minimize the squared error between the desired and predicted output. After multiple iterations the NN can make accurate forecasts.

K-Nearest Neighbour – KNN

KNN is a further popular forecasting method as shown in the literature review. It is a non-parametric method used for classification and regression problems. It does is often used as it does not make any assumptions about the underlying data distribution. It is often called a lazy learner as it does not create a model for the dataset until carrying out a query on it. When presented with an unlabelled point it estimates the likelihood of what group it belongs to based on what group the points nearest to it are in. For regression problems it simply takes the average of the datapoints for the group it is assigned to. The main challenge for this method emerges when we have a very large number of datapoints. To be able to search the dataset rapidly and find the most similar point special methods are used. For example, indexing or selecting one point to represent a group can enhance the speed of this algorithm.

Support Vector Machines – SVM

SVM is another popular machine learning model that is used in regression and classification problems. To understand how SVM regression (SVR) works we first need to understand SVM classification (SVC). Unlike other methods that try to fit a line to the dataset and minimizing the distance SVC tries to maximize it. It separates datapoints using a line, or so-called hyperplane, in a way that the distance between it and the closest points (support vectors) to it on either side is maximal. This margin is linked with generalization as the larger the margin the less chance there is for a new datapoint being misclassified. If the data is not linearly separable it maps inputs to a higher dimensional feature space using the kernel trick and distribute the data into two sets whilst trying to find the largest margin between them. If the margin for the hyperplane is very small the SVC tends to overfit the dataset. To overcome this issue the soft margin SVC was introduced where misclassification can be allowed. To limit the number of misclassifications the hinge loss is established and incorporated into the SVC cost function. Every misclassified datapoint has loss associated with it described by equation (5):

(5)

Where t is the actual outcome and y is the output of the SVM model. We can see that the loss starts at the normalised margin thus even for a correctly classified point that is close to the margin a loss incurs.

For SVR we replace the hinge loss with the regression equivalent called epsilon-insensitive loss described by the following equation (5):

(6)

Where y and t are the actual and predicted value and epsilon is the tolerance for error. Note that values inside the epsilon boundary the loss is 0.

Random Forest – RF

In Random Forest we have a collection of different regression trees. The regression trees are made up of nodes where a yes/no question is asked regarding a characteristic of the given datapoint. The datapoint is passed to the next node in the tree depending on the answer until it reaches the leaf (end) node. The final prediction is the average of the target variable of the points at the end node. After multiple iterations this tree can predict an accurate value for datapoints. Whilst this model can be applied to a variety of problems, is simple and easy to understand it has a major weakness that it tends to overfit the data. To overcome this problem the Random Forest model has been developed that averages the output of multiple decision trees. The more diverse the combined trees are the more accurate and stable is the RF model for. To ensure diversity the trees are trained by taking random samples with replacement from the training dataset until the new dataset is of the same size as the initial dataset. The generated datasets tend to have around 60% of the original dataset with the remaining being duplicates [16]. Afterwards, for each new dataset, random feature selection is introduced to maximize variety in the trees. and randomly selecting features. The Random Forest model is proved to be very fast and robust.

# Methodology and results

This section gives a detailed description of what has been done for the project and an analysis of the results. The section can be divided into 3 main subsections/stages:

1. Data pre-processing
2. Model development
3. Interpreting the model

## Stage 1 Data pre-processing

This section describes the data pre-processing stage. It will be shown how the data was explored, what information was gained and how the data was cleaned and enriched.

### Exploring the data

Given the importance of EDA as described above we start off by exploring the data and visualizing it using different plots. Plotting the HH Electricity data we get the following graph:

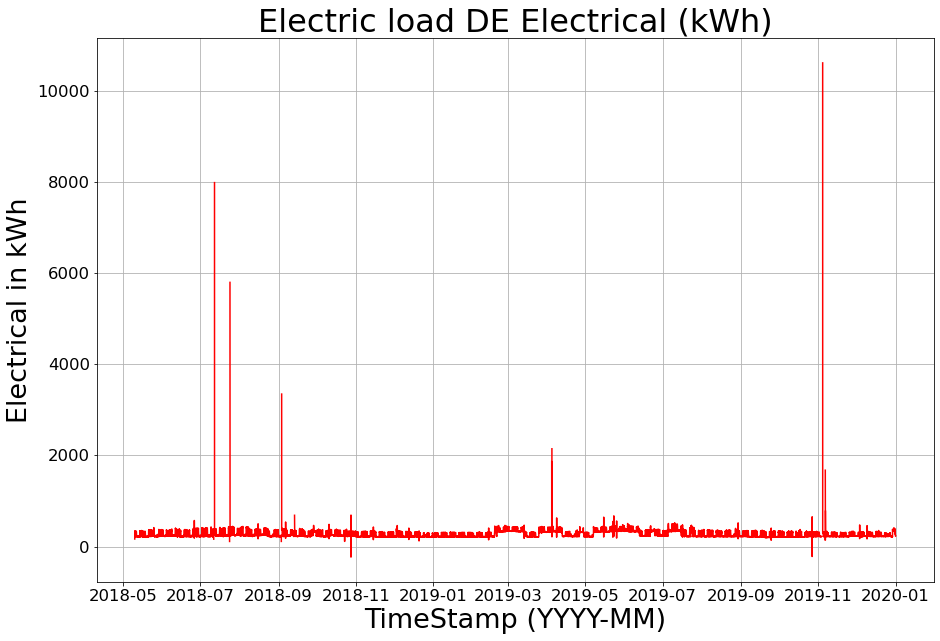
 

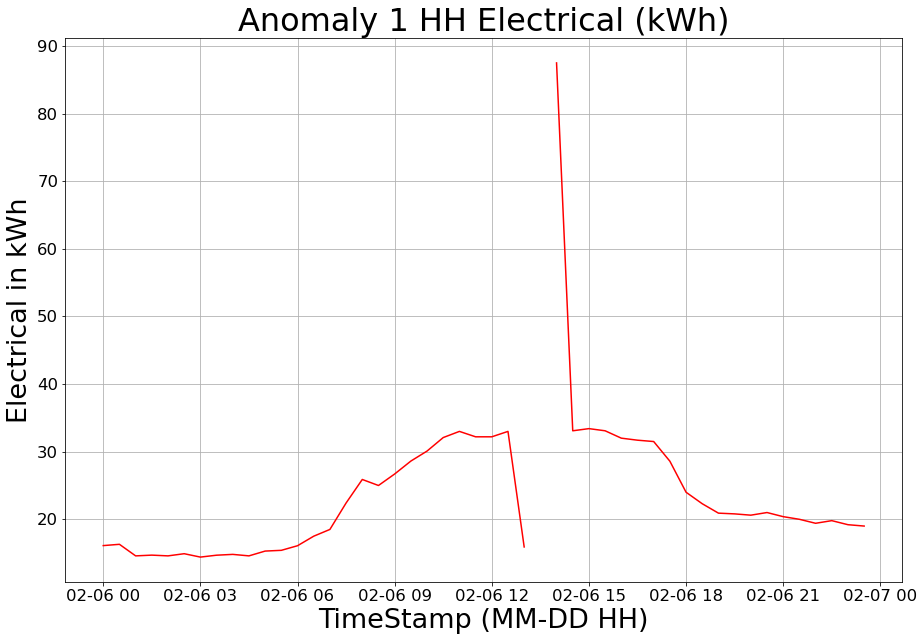
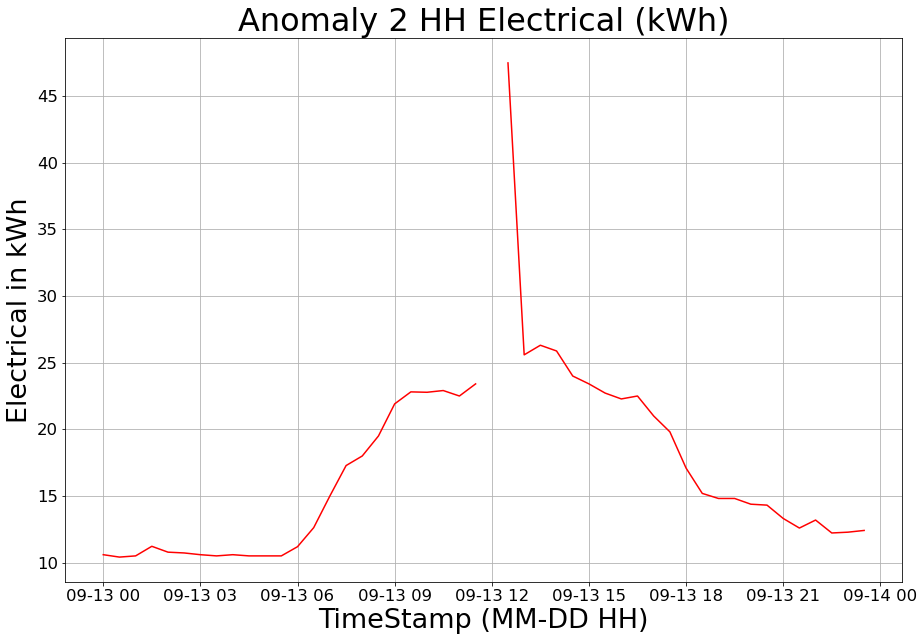
Figure 1 - A plot of HH Electrical data (left) and DE Electrical data

Figure 1 shows us the plots of the raw data. Note the difference in ranges for axes. We can see that the electricity consumption for the DE block is significantly larger compared to HH. Furthermore, we can see that whilst the timeframe for HH starts at 01/01/2018 and ends at 31/12/2019 the DE block is missing the first 5 months as it only starts at 10/5/2018.

We also notice multiple very large values that must be outliers in the dataset. Furthermore, we find negative values. Given that all datapoints are measurements of electricity consumption thus, values should be non-negative all negative values must be outliers. As outlined in the literature review and background knowledge sections outliers are very dangerous for training ML models. If an outlier like this gets into the training dataset the model would try to adjust its parameters to fit itself to the outlier. Assuming the model is trying to minimize the squared error to penalize significant misfits this might become even more dangerous. Thus, we will need to implement a way to deal with these outliers.

To get more familiar with outliers we visualize all of them in both datasets with a shorter timeframe to gain in-depth details. A few significant examples can be seen in figure 2. We investigate the clear outliers that are shorn in figure 1 and possible outliers identified using the z-score. When doing so we notice that all outliers follow one of the following patterns:

* Low value – missing value(s) – Outlier (Pattern 1)
* Missing values – outlier (Pattern 2)
* Outlier – Low value – outlier (Pattern 3)
* Low value – high value (Pattern 4)

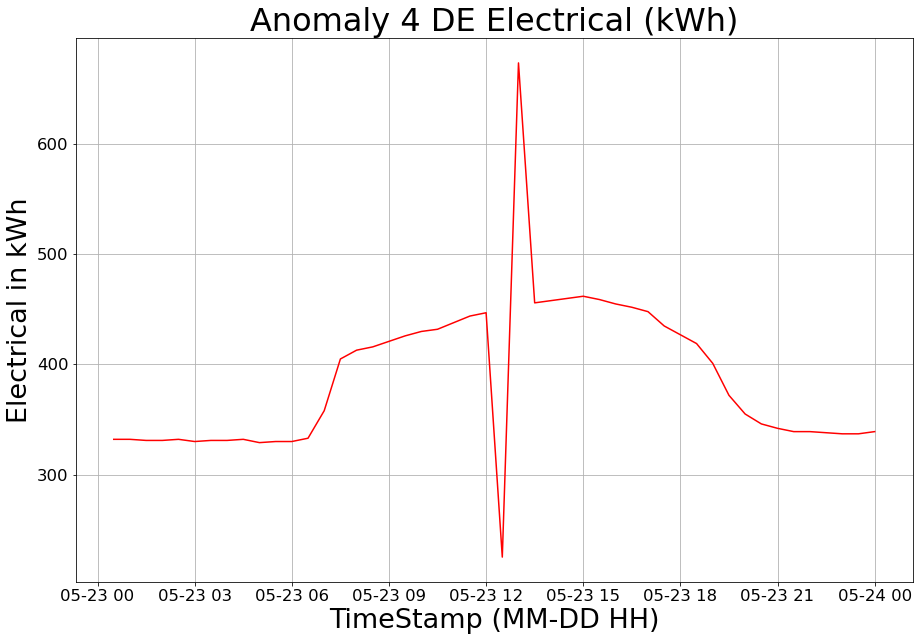
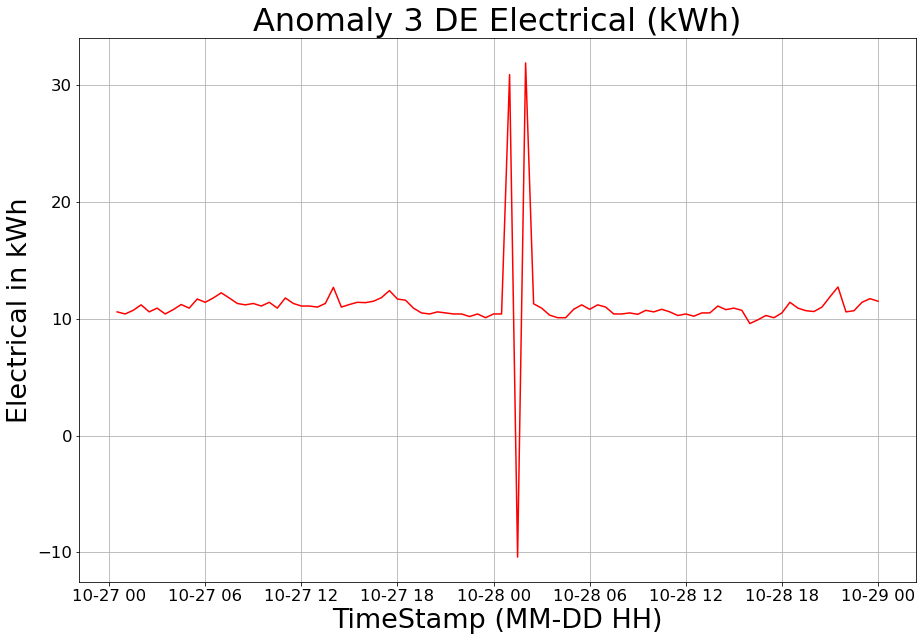
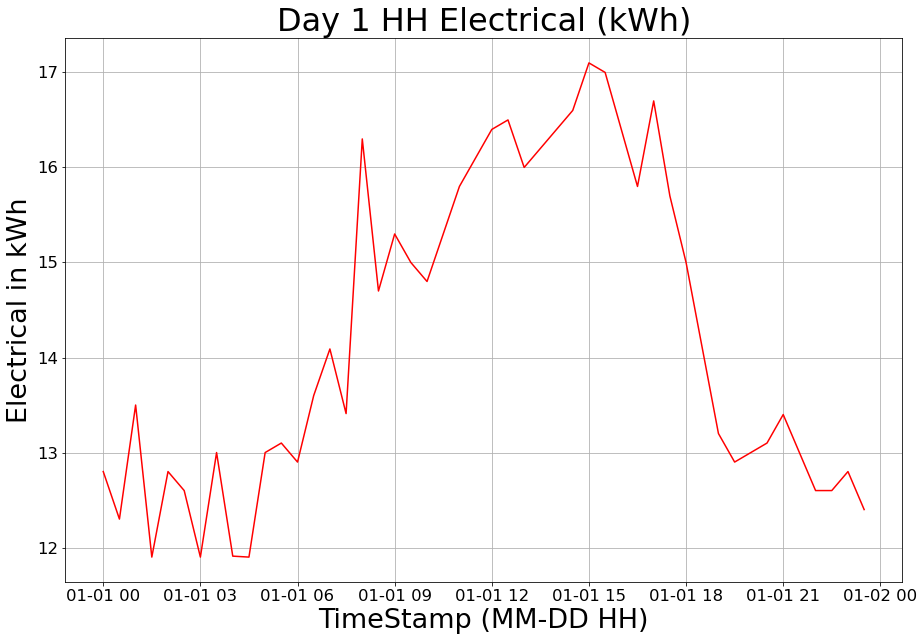
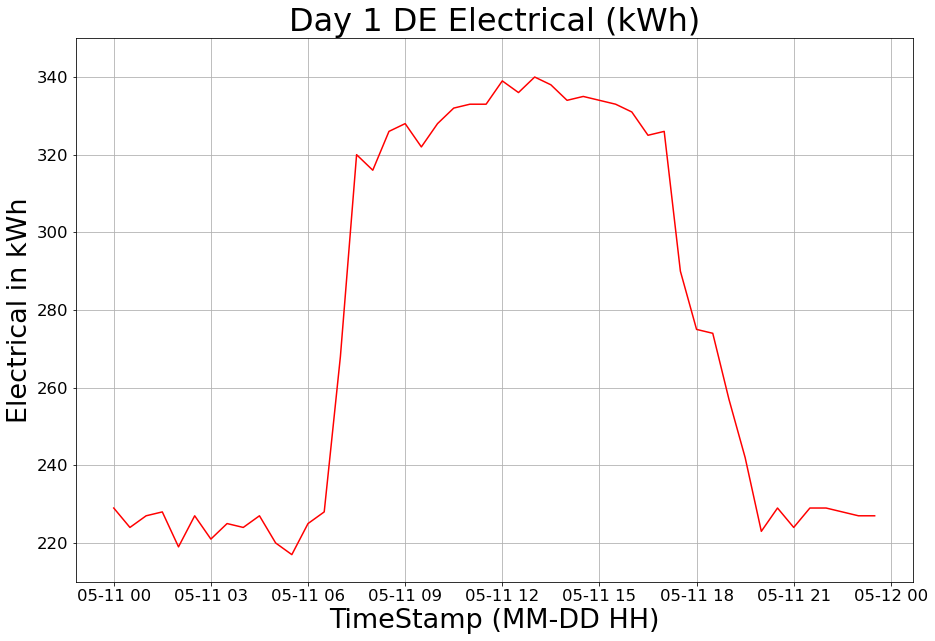
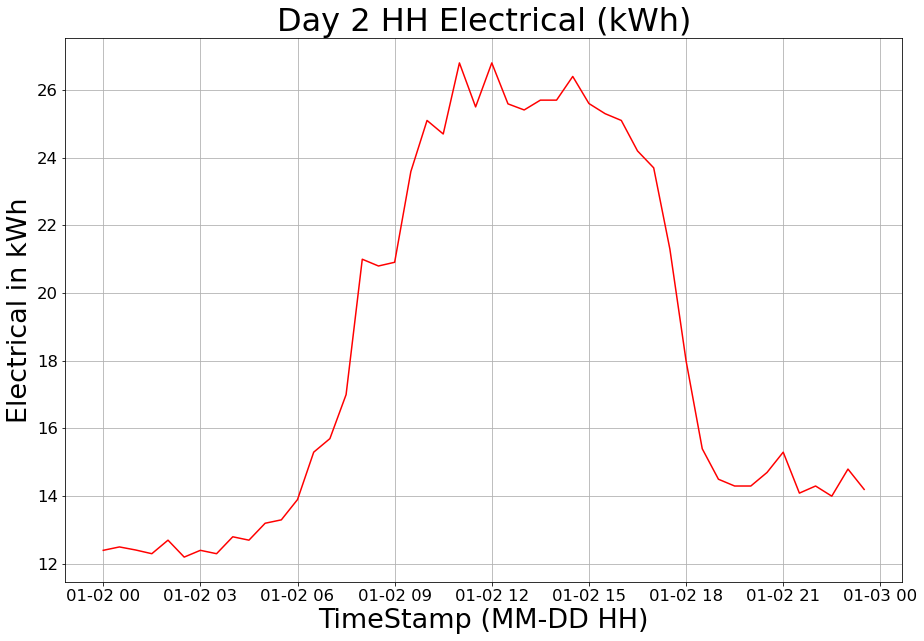
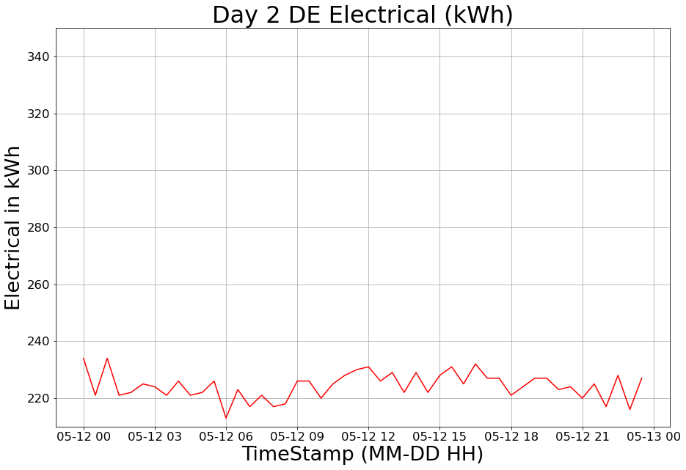


Figure 2 - Plots of the typical pattern First row left - Pattern 1, first row right - Pattern 2, second row left - Pattern 3, second row right - Pattern 4

To gain a better insight into the non-outlier parts of the data we visualize the electricity consumption for a few randomly selected days. The following figure 3 depicts the first 3 days for each dataset. Note that for DE the measurements only start at 9:30 10/5/2018 so Day 1 refers to the first full day. Furthermore, day 3 for DE was skipped as it was a weekend date with the same characteristics as day 2.

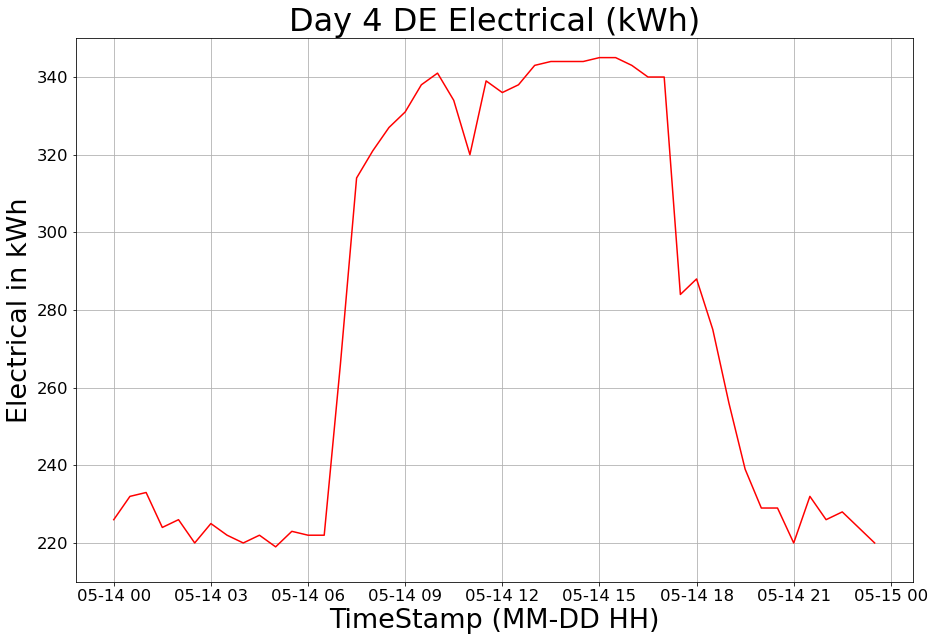
 

Figure 3 - Plots of the electricity consumption for the first days for each building block (HH left, DE right)

When inspecting randomly selected days, a further assumption was formed that normal daily usage also follows a pattern. The average day can be divided into 4 main segments:

* Starting at 5-6 consumption starts increasing rapidly until around 9-10
* From around 9-10 consumption stays high until around 16
* From around 16 consumption decreases rapidly until 20-21
* Starting at 20-21 the consumption is low until 5-6 during the night

This behaviour is as expected given that the data comes from an office building.

### Cleaning the data

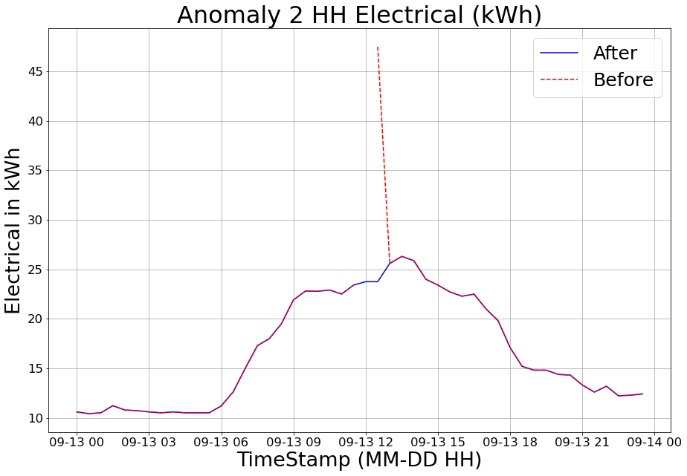
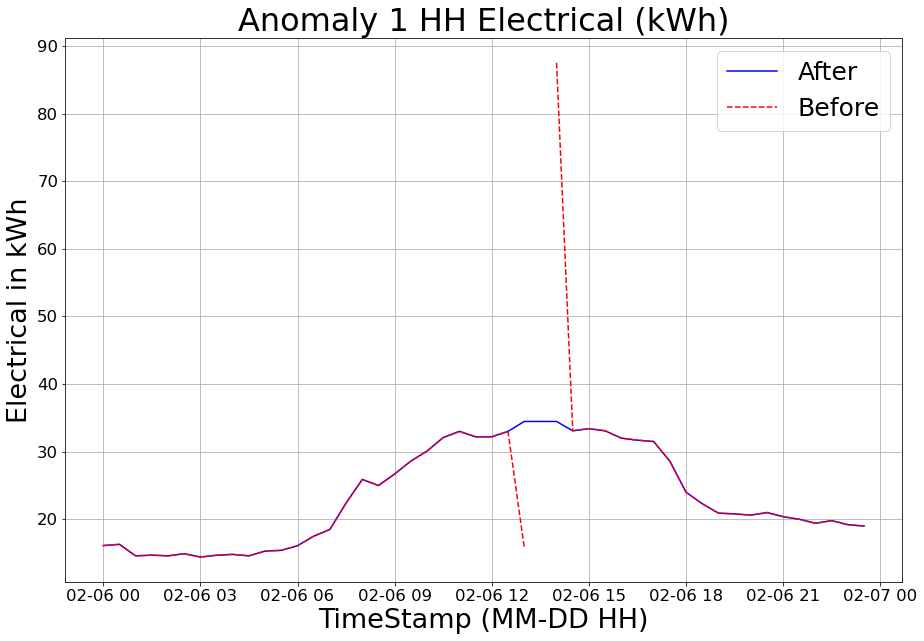
In this section we aim to clean the data from inconsistent points. For more information about the cleaning process regarding the initial differences in comparison to [9] please see Appendix A. The inconsistent points are made up of the previously identified outliers and the multiple missing points that we encounter in the dataset (excluding the missing 5 months in the DE dataset). As previously identified, outliers in the dataset follow a pattern that should be targeted during the cleaning process. As we pointed out in the literature review the most common method when dealing with outliers is removing them and for missing points to impute the mean. This would be a very bad option as it would distort the profiles of the time series data.

When further investigating the outlier patterns it was found that it seems like whenever an error occurred (for example sensor malfunction) the information was still preserved within the outlier pattern. It is found that apportioning the sum of the pattern across the affected datapoints restores the lost information and deals with the outliers as well. For missing values that are not covered by the known patterns linear interpolation has been applied. To be able to detect the patterns we need a method to detect the locally low and high values that were found in the pattern and are clear outliers given their surrounding points but were not classified as such using the z-score method. For that a new dataset was created with the differences between consecutive points using the pandas.diff() function. Now, we can calculate a maximum threshold for the difference using the maximum fence of the interquartile range.

The developed cleaning algorithm traverses through every item of the dataset and checks if there are any modifications required:

1. If it finds a missing point it analyses the following points as there could be multiple missing points which end with an outlier, indicating pattern 2 that would trigger the algorithm to apportion the sum. If it does not find an outlier at the end it only found missing points outside the pattern, thus linear interpolation is applied.
2. If the current point is a low value (taking the negative difference with the previous point and it is above the threshold) we analyse the following instances. If the following value is missing, we encountered Pattern 1 thus by repeating the previous process (1) we identify all affected points and apportion the sum. If this is not true but the following value is a high value (taking the positive difference with the previous point and it is above the threshold) we encountered Pattern 4 thus, we apportion the sum of the pattern amongst the affected values
3. Lastly if the current point is an outlier we analyse if the next two points are a low value followed by an outlier. If that’s the case, we encountered pattern 3 and we apportion the sum across the pattern

To confirm the success the algorithm we print out variable *I* whenever a modification is made, and we inspect the applied modification. We see that the algorithm only got triggered where it was justified, and it restored information perfectly except for 2 points where manual adjustments were made (during midnight hours where artificial data creation is straightforward). The final result of the cleaning algorithm is depicted below on figure 4.



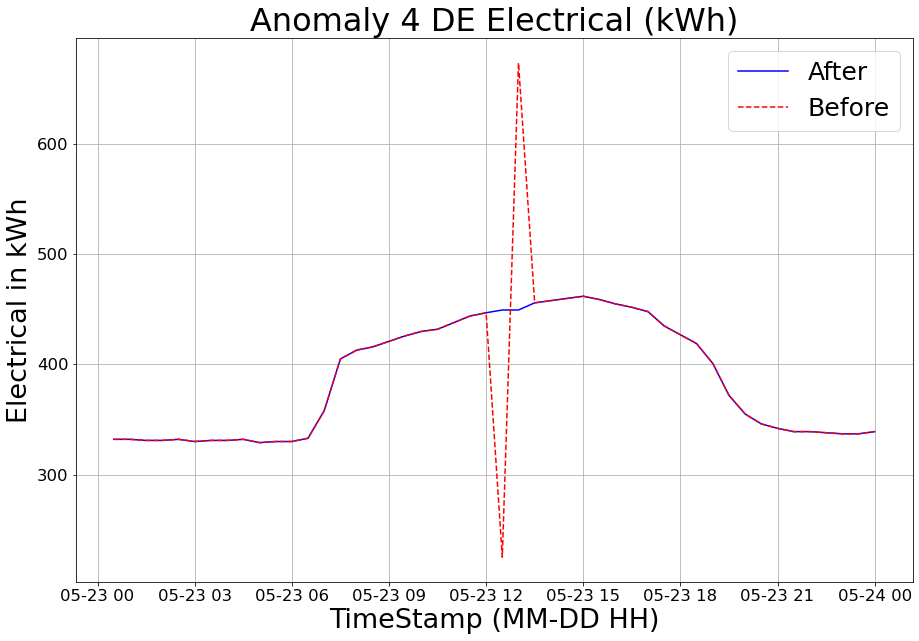


Figure 4 - Previously shown anomalous points after the cleaning process

As we can see, the previously shown outliers are all clean now with the original information restored. On figure 5 we can inspect the entire dataset without the outliers.

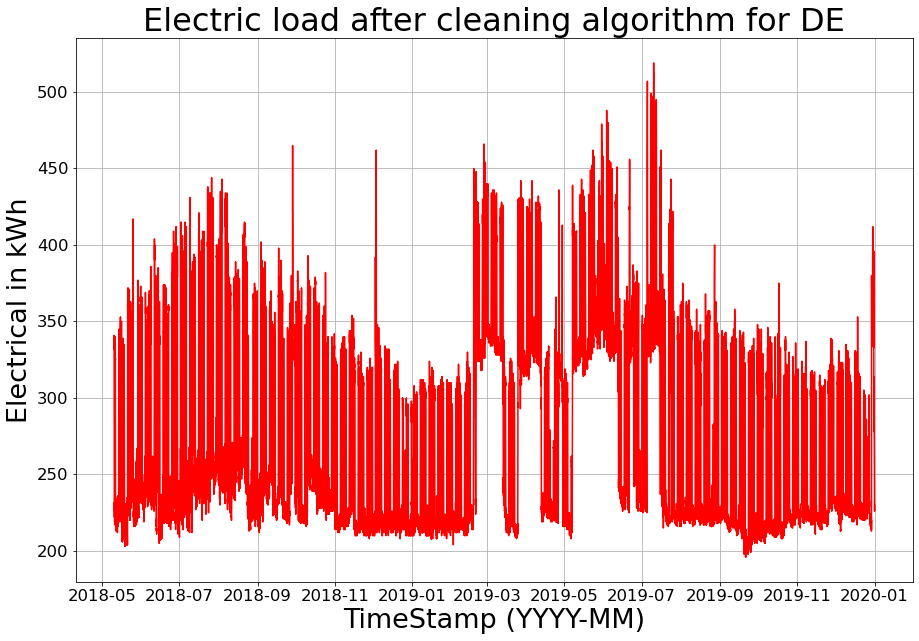
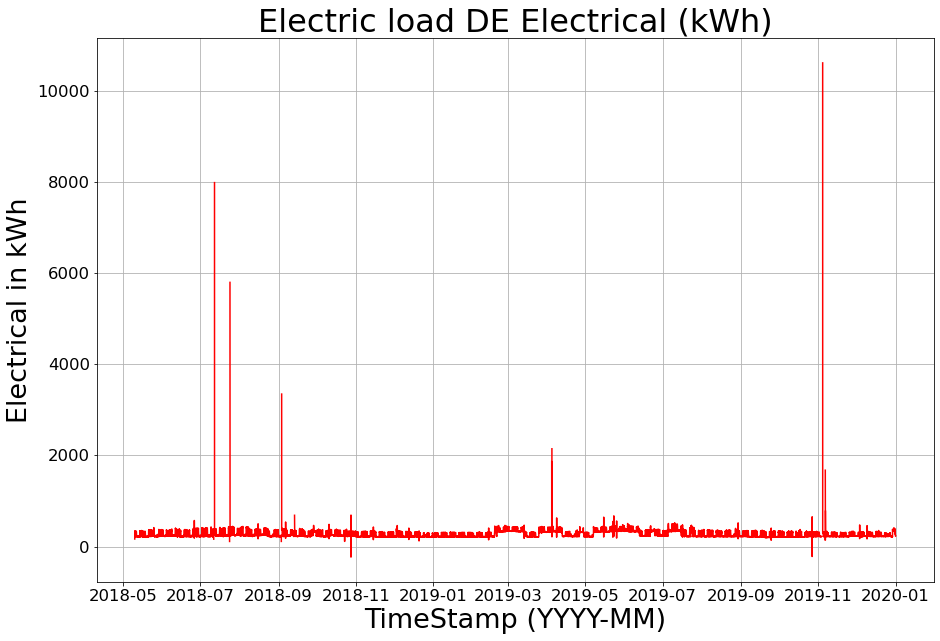
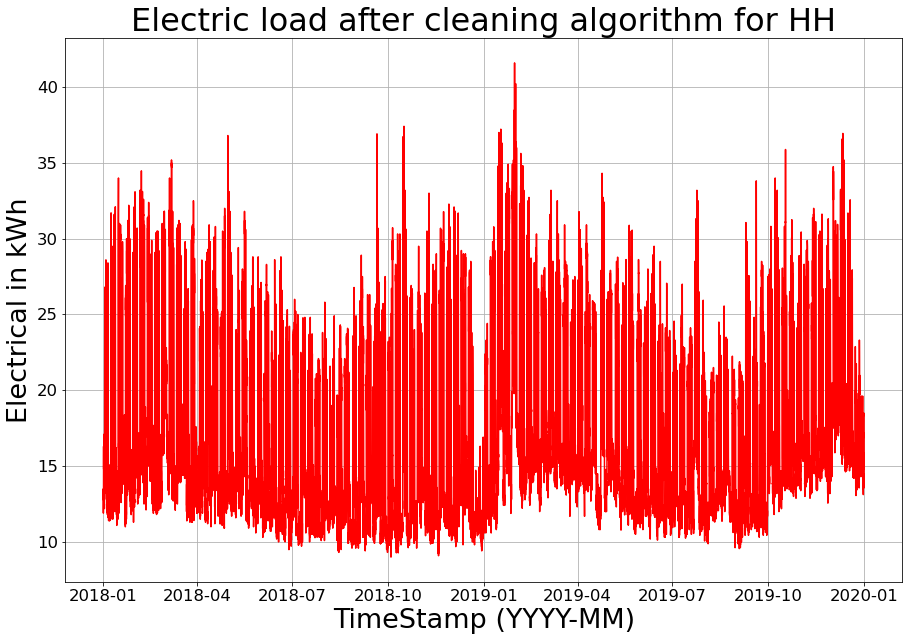


Figure 5 - Datasets before and after the cleaning process

### EDA for cleaned dataset

Once the dataset is cleaned it is beneficial to analyse it again in more depth to extract hidden knowledge which might aid the structuring of the dataset when transforming it into the final dataset. To confirm our previous assumption about days following a specific pattern depending on hour we create the following plots using the clean dataset:

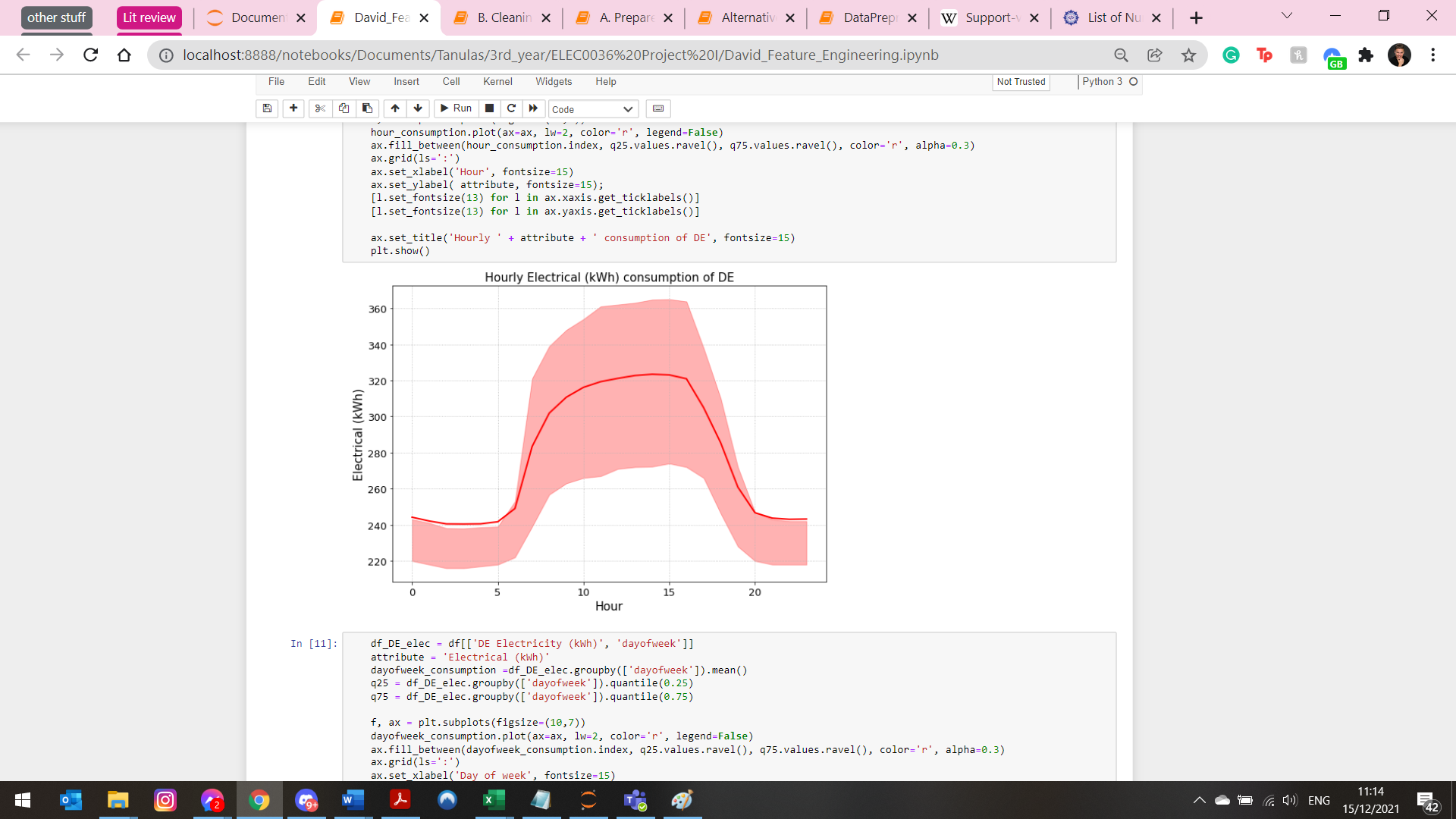
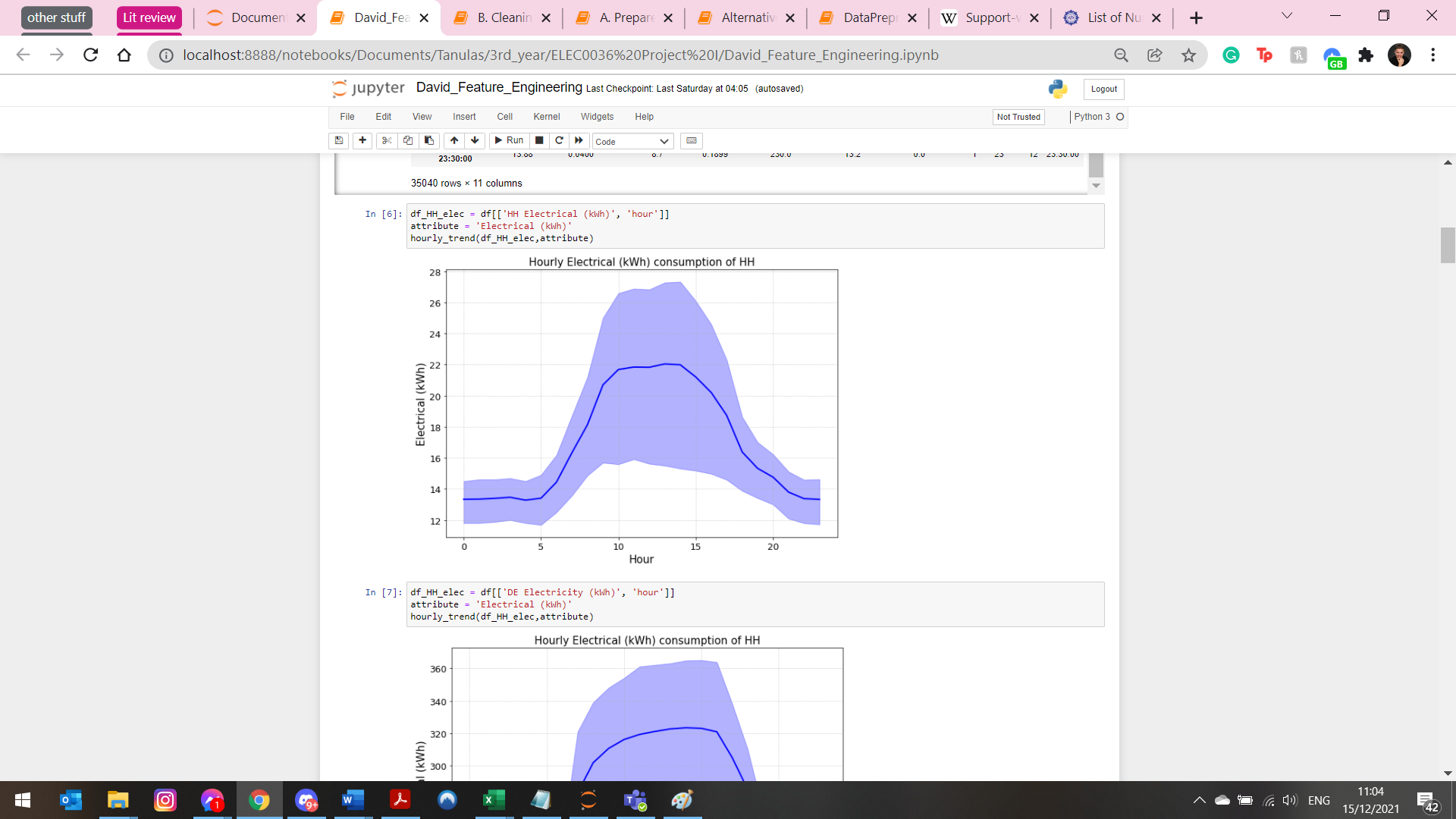


Figure 6 - Hourly electrical consumption for both building blocks (HH left, DE right) using IQR plots

The results can be seen on figure 6 and confirm out original assumption. As we can see both building blocks exhibit a significant dependence on the hour of a given day suggesting that it needs to be used as an input parameter for forecasting models.

Repeating this for day of the week on figure 7 and for months on figure 8 we get the following plots by summing energy loads for every day and grouping the days together:

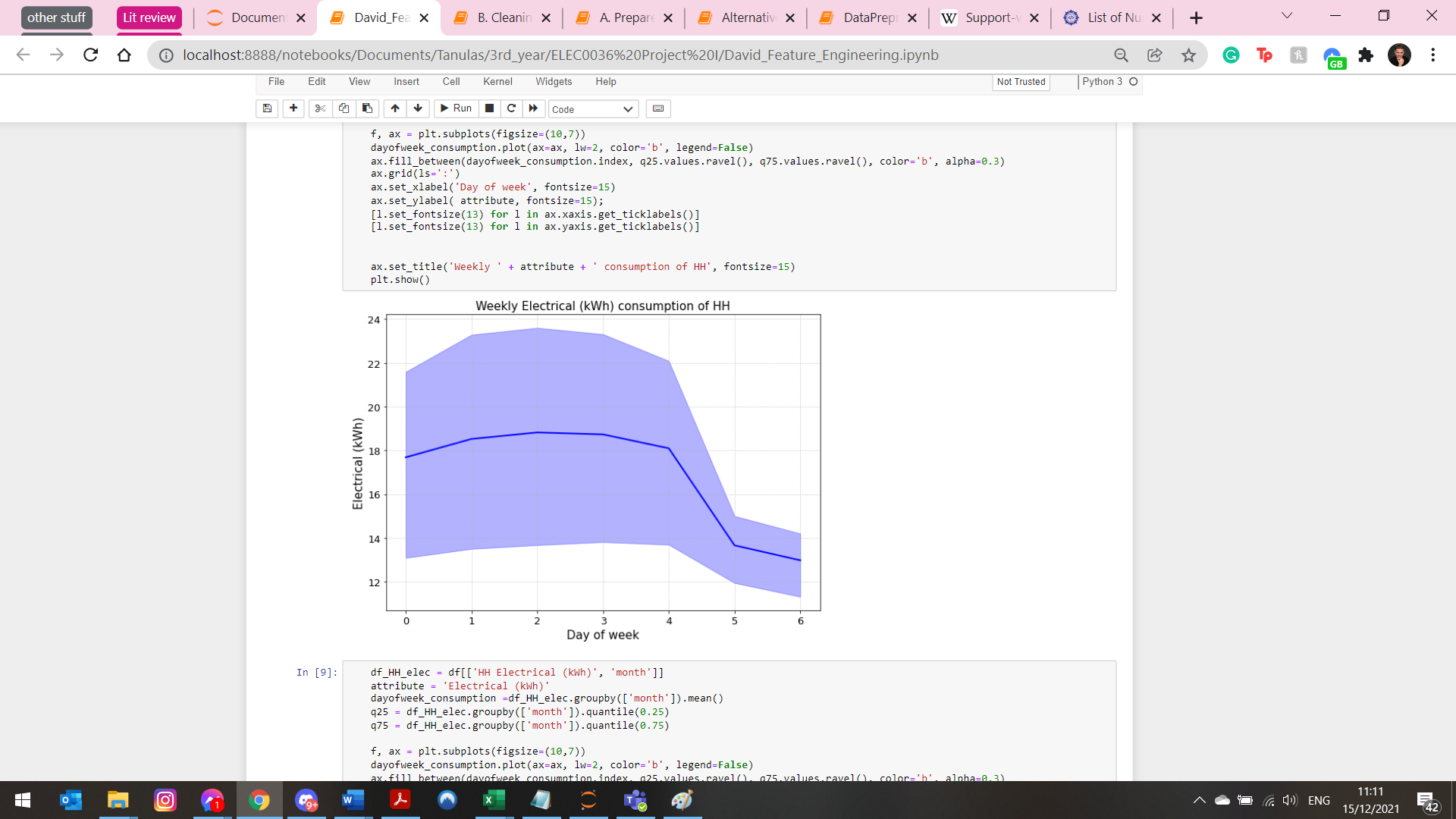


Figure 7 Weekly trend for HH (left) and DE (right)

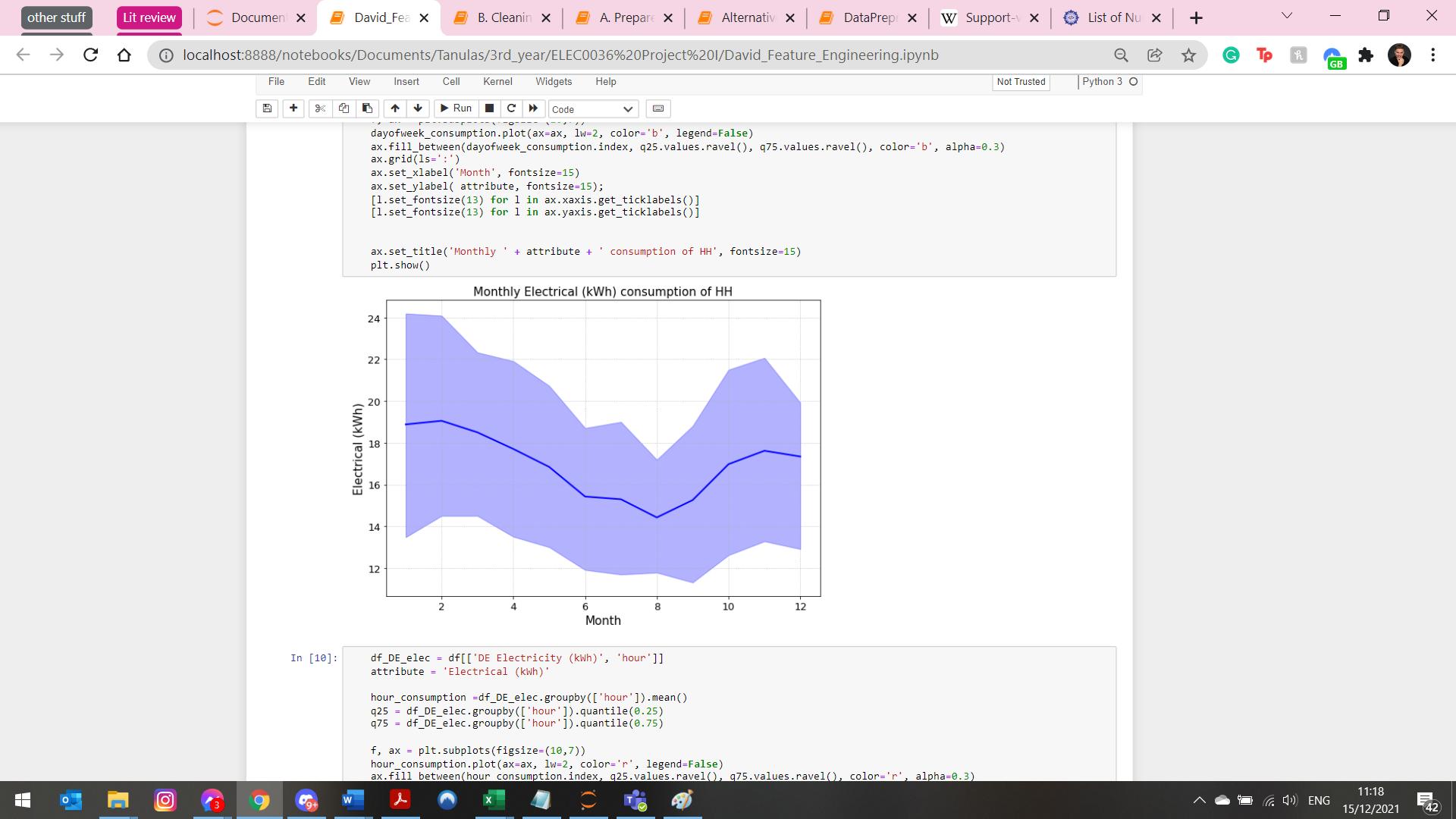


Figure 8 - monthly trend for HH Elec (left) and DE Elec (right)

As Figure 7 suggests, the day of the week is also an important predictor for energy load forecasting. Whilst seeing obvious trends on figure 8 it is interesting to note that the two trends are reversed. For HH the electrical consumption drops during summer due to decreased heating energy demand whilst for DE block it increases. This can be explained by the DE building being more modern and having an HVAC system whilst cooling in the HH building block relies on natural aeration.

### Data enrichment

To allow for more accurate predictions as shown in the literature review, we will enrich our dataset with additional exogenous factors. The occupancy data of the buildings are unavailable but outdoors weather conditions can be obtained from nearby weather stations. Unfortunately, the dataset used in the work of Kalliga, Polyxeni [9] became unavailable, but a new source was found that can supply weather data from a nearby station. The dataset was purchased and downloaded from visualcrossing.com. To confirm the correctness of the data EDA steps were performed. The dataset consists of values that describe current weather conditions such as humidity, outdoor temperature, and others. The time interval between measurements is 1 hour thus showing changes in temperatures throughout the day. As there is a mismatch between datasets due to having different time intervals between measurements the weather dataset was converted to a data frame with half hourly measurements by duplicating each row.

## Model development

This section describes each part of the model development process. A flowchart for the process can be seen in figure 9 where the blue rectangles describe the model development process.

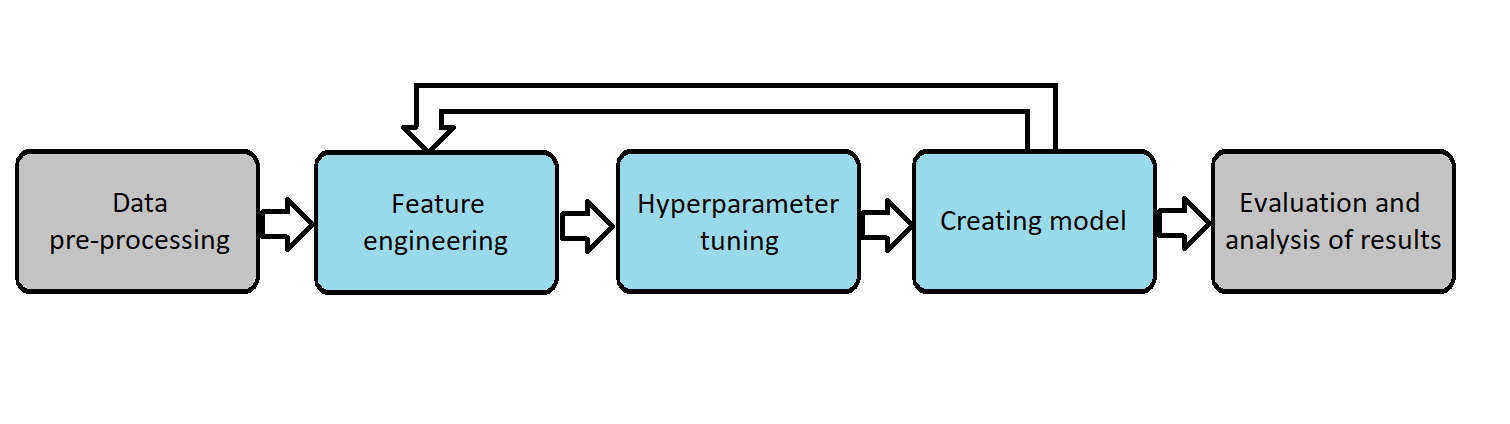


Figure 9 Flowchart for model development process model development part highlighted in blue

### Feature engineering

Feature engineering is used to improve the accuracy of machine learning models by creating new features or selecting a subset of features from the dataset using domain knowledge. It is a repetitive part of the feature engineering process as there is a variety of possibilities when engineering features. As indicated by the flowchart on figure 9 when engineering a set of features it is then tested on the dataset by developing a model using that set of features. If the results are unsatisfactory the process is repeated with different features. Creation and selection of features is based on literature review or intuition given the knowledge extracted from the dataset.

Based on the knowledge extracted from the EDA process the following features have been created using the time and holiday libraries in python:

* Hour (the hour when the measurement was made ranging from 0-23.5)
* Dayofweek (the day of the week of the measurement ranging from 0 (Monday) to 6 (Sunday)
* Month (the month of the measurement ranging from 0 (Jan.) to 11 (Dec.)
* Holiday (0 if day of measurement was not holiday, 1 if it was a holiday)

For the feature selection process, multiple combination of the available features has been experimented with. It was mostly guided by combinations in literature review but also with multiple experiments based on own ideas. The most common combination in literature review were the following:

For historical values:

* using a few of the immediate past values to make one step ahead forecasts and repeat the forecasting step by step until the desired frame is forecasted.
* Using the value from 24 hours ago
* Using all or parts of the values from the past day

For exogenous features

* Outside temperature
* Humidity
* Occupancy schedule
* Different combinations of these

When experimenting with the number of immediate past values to be used for one step ahead forecasts it was found that they perform terribly when repeating the steps using artificially predicted values. Despite S. Karatasou [15] showing that step by step forecasts can outperform methods where all values from the past day are used the performance for this dataset was very poor no matter the number of past immediate values used. When assessing the feature importance of these models I found that predictions are dominated by the very last value for all models. Despite the accuracy for one step being very satisfactory even the smallest error was rapidly amplified by each step.

Thus, I started experimenting with past values whilst skipping immediate values. When implementing ideas from the literature review, using all values from the past day was the most accurate but the large number of features introduced a very long computational time. When assessing that model, it was found that out of all values from the past day the one that was exactly one day before had the largest importance.

Given that our dataset comes from an office building where work has a weekly schedule the idea emerged to use values from the past week instead of the past day. Using the past day can be very unhelpful for example on Mondays where the previous value was a weekend day thus, minimal. Using the value, a week before the unlabelled datapoint would overcome this issue. To limit the computation time only the seven values from the past week were used that were an integer number of days before the datapoint in question.

For the exogenous factors it was found that using the outside temperature (in degree Celsius), humidity (in %) and a so-called “Feels like temperature” (in degree Celsius) yielded the best performance. Whilst outdoor temperature and feels like temperature seem to be identical, they are not. The outdoor temperature is the temperature measured in shadow using a thermometer the ‘feels like temperature’ is calculated by combining multiple weather conditions such as wind or solar intensity, if it is raining or not etc.

After extensive experimentation and multiple iterations of feature engineering the following model was proposed that is depicted in figure 10. Note that to have n-7day feature for all points in the dataset the target variable was shifted by 7 days thus, leaving the first 7 days with NaN values. As a result, the first week of the dataset needed to be removed. Furthermore, before using the dataset for training the values need to be normalized. The code for generating the dataset and completing the preliminary steps for model training can be found in Appendix B.

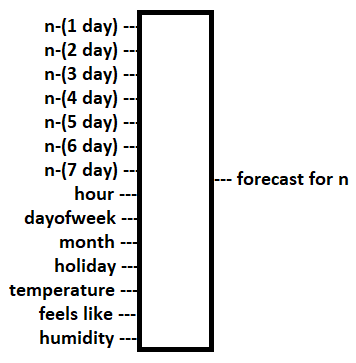


Figure 10 final inputs and output of the final model after extensive feature engineering

### Hyperparameter tuning

To maximize the accuracy of the proposed models tuning their hyperparameter has great significance. The main challenge when performing hyperparameter tuning is overfitting the dataset. To avoid this, 20% of the dataset was put away as test set for the evaluation of the model as it was done in most of the literature. Splitting the data manually can introduce bias into the evaluation as the entire test set might be made up of only 1 season thus, not representing the dataset. This result might be better or worse than the result of a random split depending on the strengths and weaknesses of the model and the characteristics of the manually selected test set. On the other hand, when splitting data randomly it is hard to visualize results. It had been decided to perform both and it turned out that there were no changes in the hyperparameter tuning process. As a result of this we will use the random split score as the evaluation metrics as it represents the performance of the model for general unseen data, but we inspect the performance using a manual split.

To tackle overfitting a further method called 5-fold cross validation (CV) was used on the training dataset to select the hyperparameters. The generally accepted number of folds is 5 to 10 times. Given our large dataset and the long computation time 5-fold CV was chosen. 5-fold cross validation means that we partition the dataset into 5 chunks and evaluate the model 5 times where for each time a different chunk is taken for testing and whilst the remaining ones are used for training. The final score is the average performance achieved over the 5 times. This ensures that the results are not biased/overfitted due to the same reasons that applied when doing a manual split for test set creation.

In the remaining part of this section, we give an in-depth description on how the hyperparameters were tuned for each model using the popular ML library sklearn. For all following methods their respective model from the sklearn library was used.

1. RF

As discussed in the background theory, the RF model tries to combine the output of multiple decision trees whilst trying to maximize their diversity. This can be done by adjusting the n\_estimators which controls the number of trees in the forest and the max\_features parameter that controls the random feature selection. As the max\_features parameter has an ‘auto’ function this did not need any tuning. To determine the best parameter for the number of trees GridSearchCV was used with 5-fold CV. Values from 10 to 1010 have been searched and the best parameter was 810 for HH model

1. SVR

For SVR the main parameters to be concerned with are the kernel function, gamma, C, and epsilon. As described in the background theory the kernel is used to map inputs to a higher dimensionality feature space. The gamma parameter is used as a coefficient in the kernel function. The two options one can set it are equivalent for normalized data thus, we did not need to tune that parameter. C is a trade off parameter that controls the width of the margin and the tolerance for error. Furthermore, as introduced in the background theory epsilon is part of the epsilon insensitive function and server as a margin for which datapoints are not associated with a loss. The hyperparameters were tuned using 5-fold CV GridSearchCV over the parameters for C ranging from to (during the feature engineering iterations 1000 has been removed due to producing poor results but requiring long computational time. For parameter epsilon the values ranged from to . And for the kernel function rbf and linear were explored.

1. KNN

As introduced in the background section KNN Regression is a model that tries to group datapoints based on their similarity to neighbours. For this method we only need to tune the number of neighbours that are used for kneighbour queries and the weight parameter that is used in the weight function when making predictions. The algorithm that is used for calculating the nearest neighbour has an auto function thus, not requiring parameter tuning.

The n\_neighbors parameter has been tuned in range of 1 to 150 with uniform and distance weight thus, totalling in 300 fits.

1. ANN

For ANN, the multi-layer perceptron was used from the sklearn.neural\_network library. The hyperparameters to set for this method are hidden layer size that specifies the number of neurons in the hidden layers in range (1 to 35), the solver that is calculating the weights at nodes(lbfg and adam), the max\_iter that specifies the number of maximum iteration where it stops even if the model has not converged yet (3000), the learningrate\_init that specifies the initial stepsize (it decreases with each step) when adjusting weights (10.0 \*\* -np.arange(1, 6, 2)).

The best parameter combination for all models can be seen in the following table:

|  |  |  |
| --- | --- | --- |
| Model | HH best combination | DE best combination |
| Random Forest | Trees = 810 | Trees = 1510 (search range extended after plotting results for 10-1010 range) |
| SVR | C = 1, epsilon = 0.1, kernel = rbf | C = 100, epsilon = 1, kernel = rbf |
| KNN | Nneighbors = 100, weights = distance | Nneighbors = 106, weights = distance |
| ANN | LayerSize = 5, solver = lbfgs, learnRate = 0.1 | LayerSize = 11, solver = adam, learnRate = 0.001 |

### Results and analysis

The results of the HH models on the test set was the following:

|  |  |  |  |
| --- | --- | --- | --- |
| Model Manual Split | MAPE score | Model Random Split | MAPE score |
| Random Forest | 9.36% | Random Forest | 5.3% |
| KNN | 10.96% | KNN | 7.65% |
| SVM | 9.98% | SVM | 7.36% |
| MLP | 9.88% | MLP | 8.6% |

As stated in the previous section we are focused on the MAPE scores for random split results as they are more representative of the performance of the model. Furthermore, as we can see all models performed similarly poorly on the manually split data indicating that it was biased towards being less predictable. Given the Random Split MAPE scores Random Forest is a clear winner with 5.3% MAPE accuracy when labelling unseen data.

The results for DE models on the test set are as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| Model Manual Split | MAPE score | Model Random Split | MAPE score |
| Random Forest | 5.34% | Random Forest | 4.01% |
| KNN | 4.47% | KNN | 4.12% |
| SVM | 2.873% | SVM | 2.874% |
| MLP | 3.6% | MLP | 4.99% |

For the DE test set we can see that manual and random split scores are very similar. We also have a clear winner that it SVM with 2.87% MAPE score predicting values for unseen datapoints.

It is interesting to note that for the two building blocks that are located next to each other we ended up having different models outperform the other ones. As stated in the literature review where papers were contradicting each other regarding the models being superior in comparison to the other, this is mainly dataset dependant. These results prove that point by showing that for 2 similar datasets we found 2 different models being superior. This emphasises the importance of exploring multiple models and not to limit ourselves to 1 model.

The reason for having 2 different winners lies in the fact that whilst the datasets might be similar the profile of the buildings are very different. As we described before, HH is a low-tech building whilst DE is a high tech one. We also showed that they have different responses to the weather during the summertime.

To gain a deeper insight into the working of the models we will now look at visualizations on the manually split datasets. Please bear in mind that for HH the results for manual splitting were rather poor.

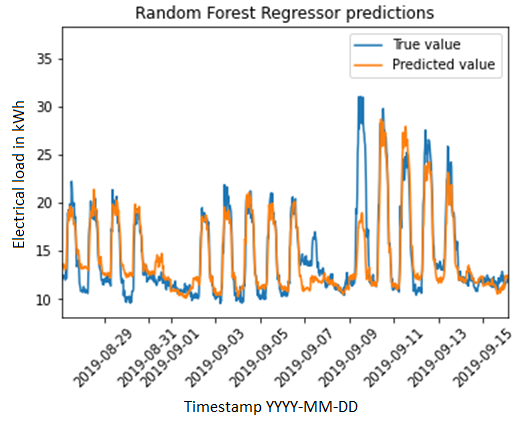


Figure 11 Interesting timeframes for HH

When visualizing the predictions for the manual test set, we find the interesting timeframe as can be seen on Figure 11 left. As we can see, the model has difficulties making accurate predictions. It is interesting to note that there are random peaks reaching 25 kWh on the weekend 7-8/12/2019. 25 kWh consumption is usually typical for working hours on workdays. Such a high value is not seen anywhere else in the dataset for weekends. When inspecting the dataset there is no indication of this rather strange behaviour thus, the model fails to predict accurately. It is also interesting to note that the following weekend the predictions try to mimic the previous “strange” weekend due to being influenced by the “n-7days” feature.

The figure on the right demonstrates multiple days where the consumption does not even reach 25 kWh during peak hours. We also see an interesting shift where the amplitude of the peak values in comparison to the off-time values doubles from the previous week. We can see that for the next day the predictions are accurate again. This demonstrates the importance of the n-1day feature in the model.

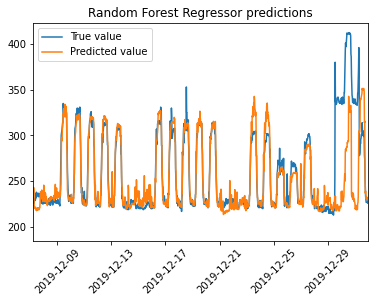


Figure 12 - Interesting timeframe SVM DE

Figure 12 show us an interesting timeframe for the DE prediction. This timeframe is interesting as it is the one with worst predictions across all timeframes. For all other timeframes the predictions are like the ones observed in the first two weeks where it is mostly matching the curves with a few random peaks missing that cannot be explained when inspecting the data. We also note the small misfit in the first two day of the Christmas week where we predicted more then the actual values. We can observe that the model was successful even at the Christmas days and shows us the effect of the holiday feature. What is surprising and unexplainable on the other hand it the behaviour on the last two days. Given that there is nothing like this in the dataset it cannot be considered as the model’s fault.

To get a clearer picture of the inside working of the model and confirming our assumptions made above about feature importance the SHAP interpretability technique has been introduced. It assesses the effects of specific values for specific features using game theory and interprets the model on a local and on a global level.

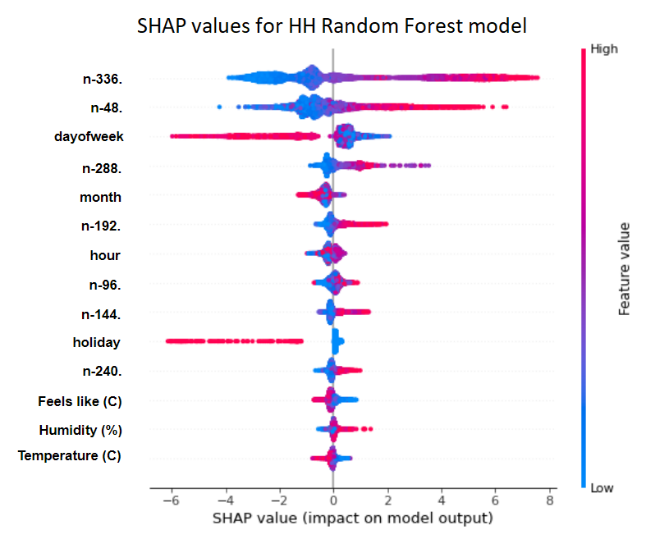
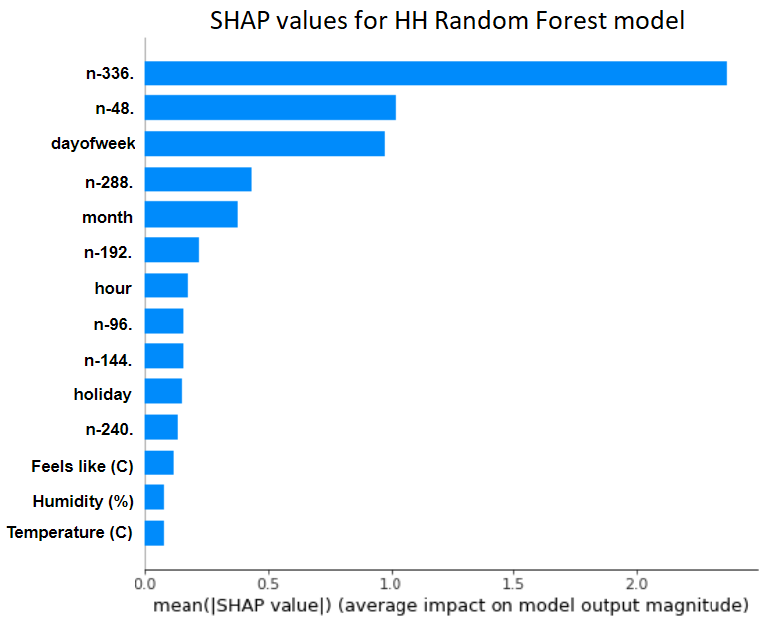


Figure 13 Interpreting the HH RF model globally

Figure 13 shows the SHAP interpretation of the HH RF model on a global level. The figure on the right depicts the importance of each feature and it confirms our previous assumptions made with n-7days and n-1days having great importance. On the figure on the right, we see the effect that values of a feature had. We can see that for all historical values a low value had a negative effect on the model as we would expect. We can also see how weekend day had a significant negative effect in the dayofweek variable. Furthermore, we can observe that the holiday feature whilst having no significant effect on the model with low (0) values it had a significant negative effect when the value was high (1). We also note how the weather conditions had very little effect when making predictions. This indicates a potential weakness that needs to be targeted in future research.

The following figures depict randomly selected points with local interpretations:

Accurate Monday 13:30

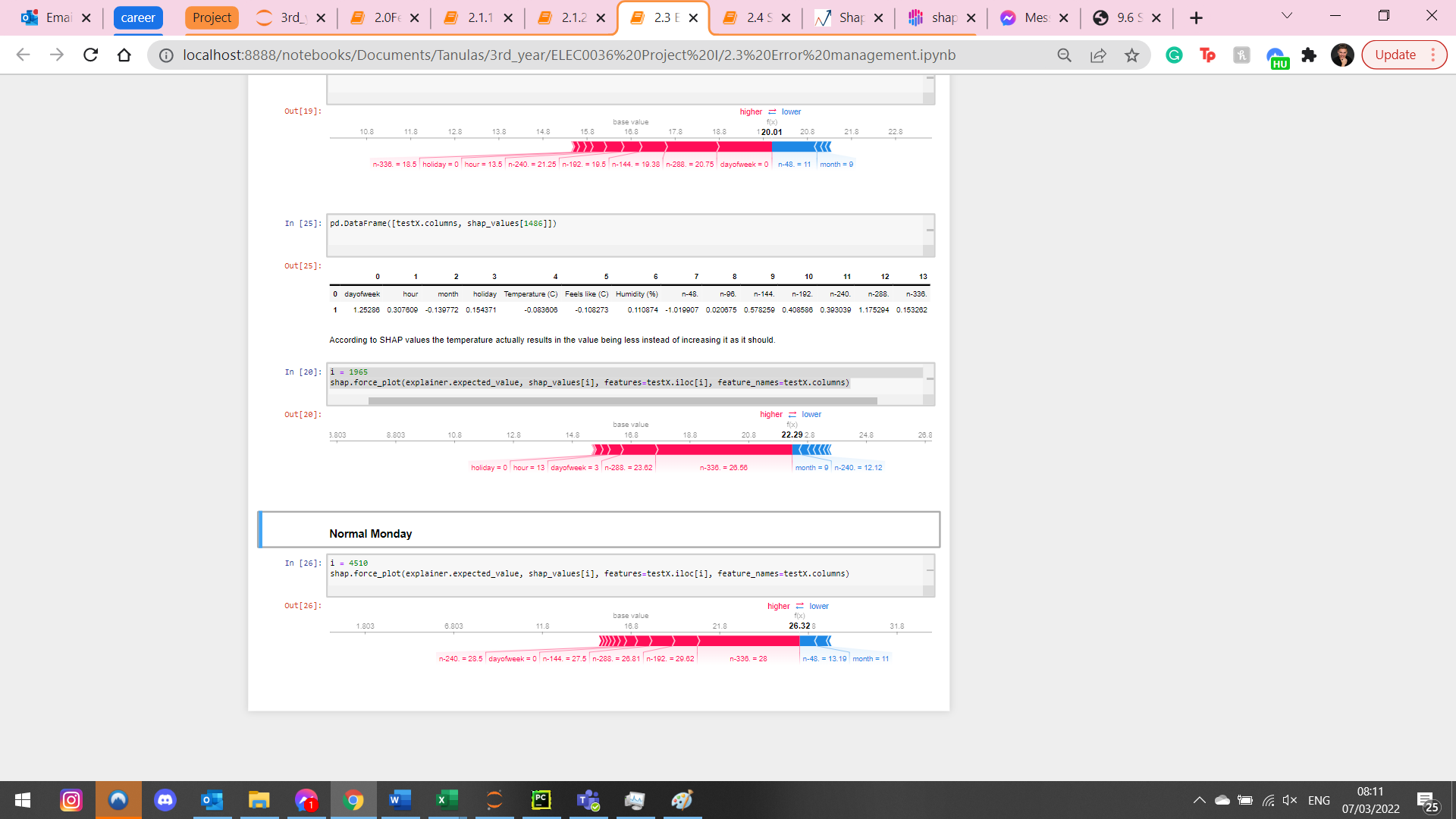


Figure 14 local interpretation of an accurate forecast on a monday 13:30

Accurate Wednesday 1:00

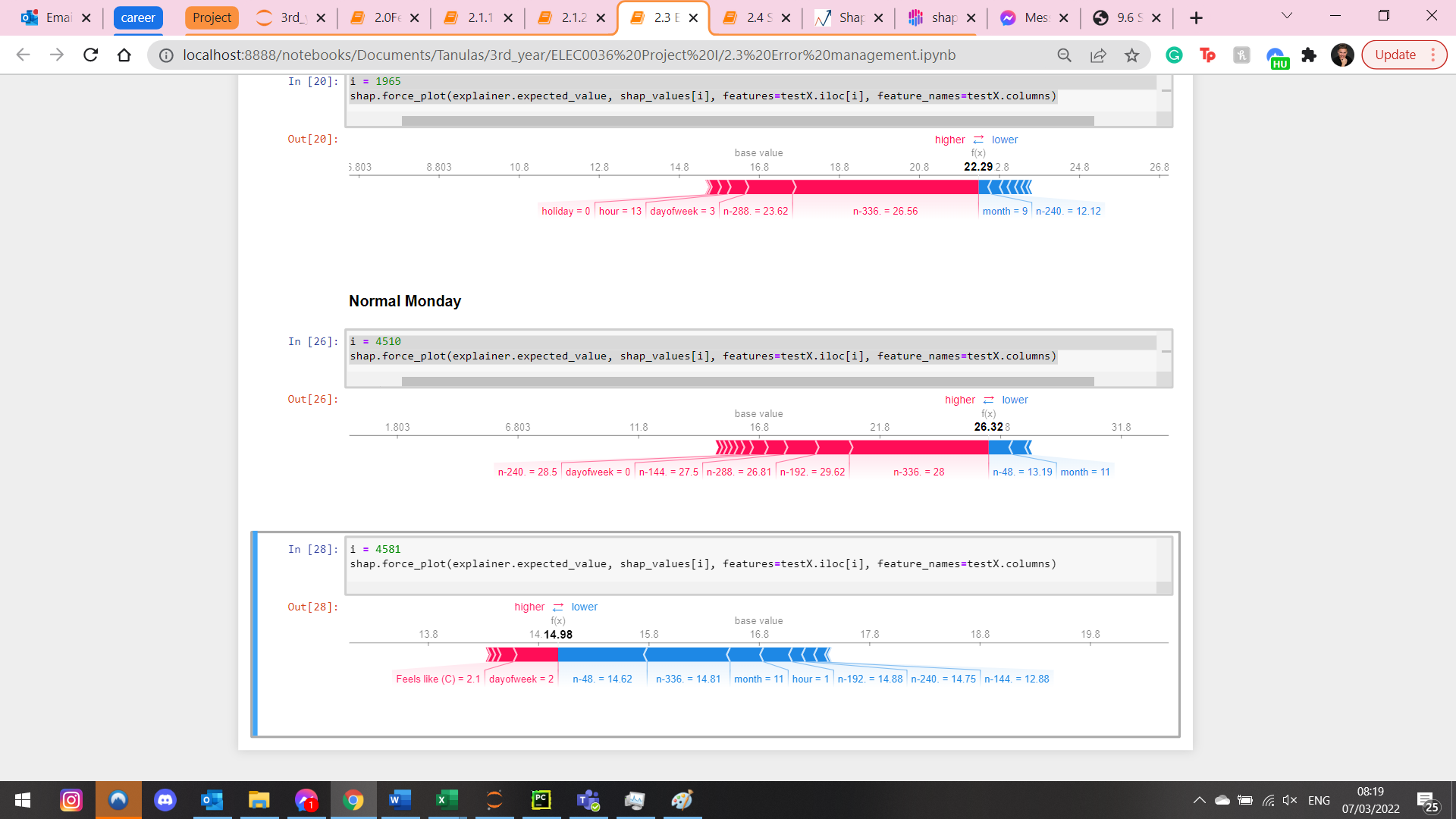


Figure 15 Local interpretation of an accurate Wednesday 01:00

Accurate Sept Weekend

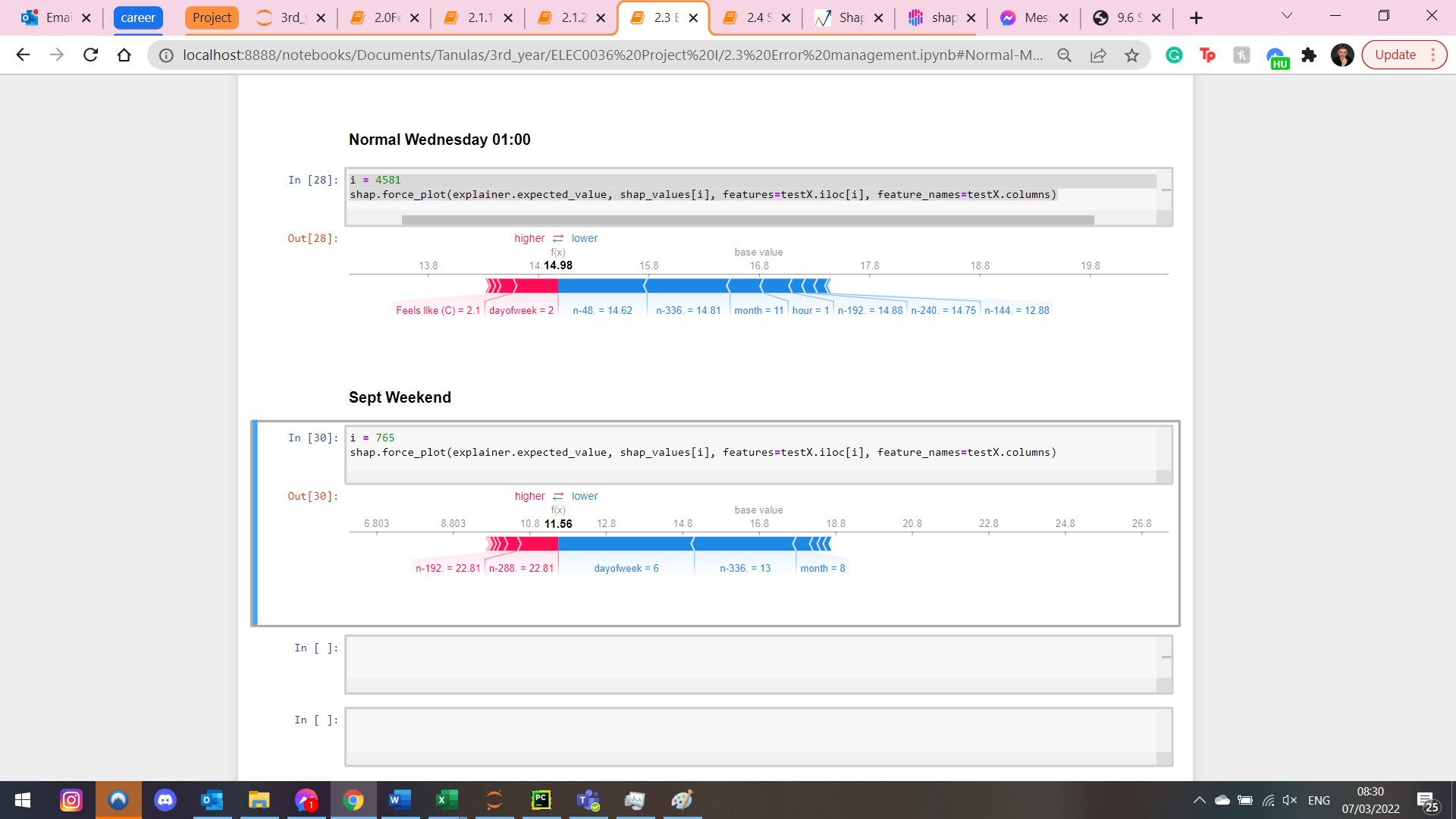


Figure 16 Local interpretation of an accurate September Weekend

When analysing Figures 14-16 we see how each feature has shifted the predicted value from the base value of the SHAP model. When inspecting the significances of the features we can confirm that our model works as expected. For the sake of interest some points have been selected where the model made a mistake.

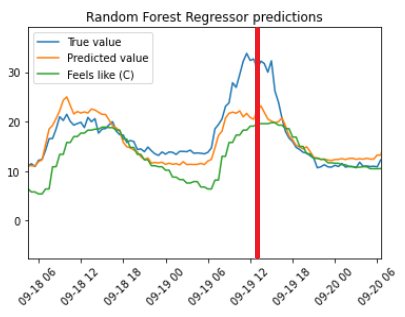


Figure 17 point that is interpreted in figure 18

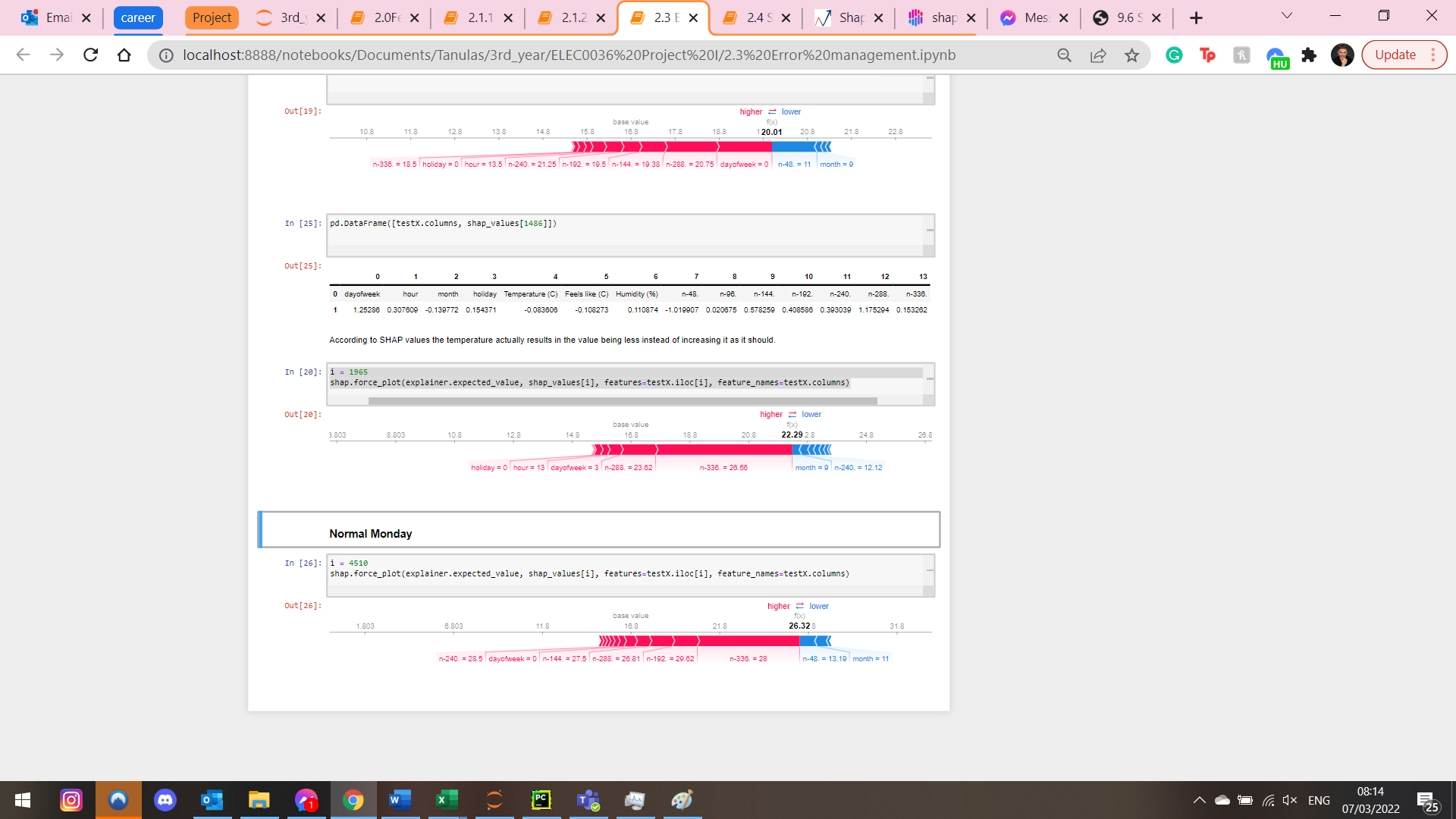


Figure 18 Local interpretation of the point from fig 17

Figures 17 and 18 depict a point previously identified as having a sudden unexplainable spike. The interpretation shows that it made the decision as it would logically make sense.

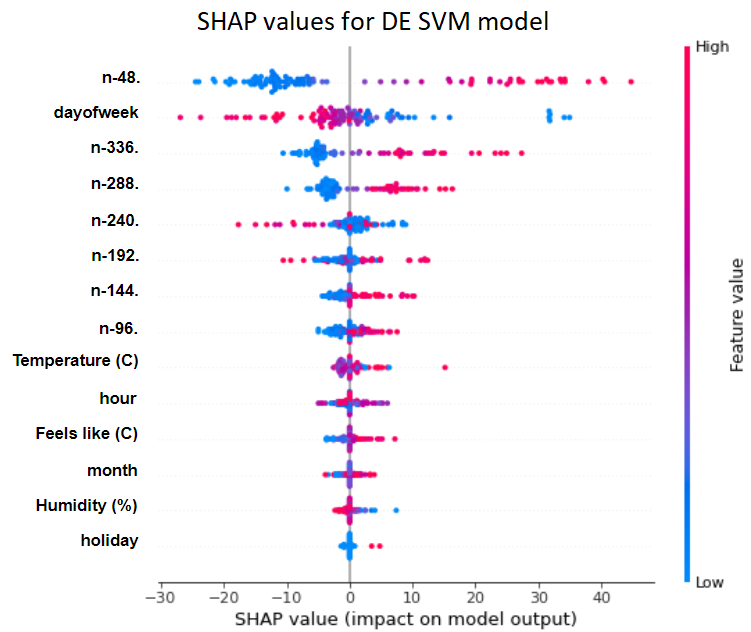
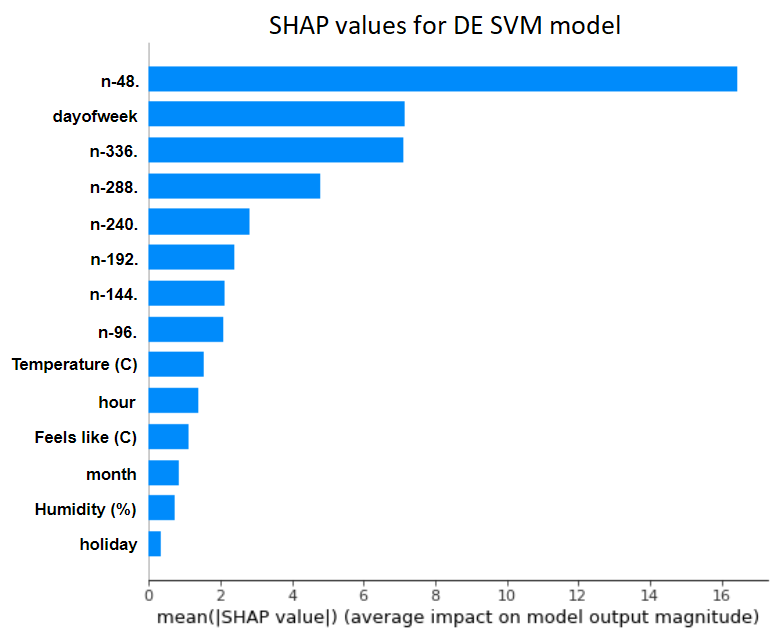


Figure 19 SHAP interpretation globally for DE

Figure 19 depicts the global interpretation of the DE model. The SHAP model was only built on a few randomly sampled points as the SHAP explainer for SVM has very high computational requirements. On the left figure we can observe how the feature importance for this model are different from the HH model. The most significant feature here is the n-1day with the dayofweek feature ranking second afterwards with a similar importance as the day-7 feature. The figure on the right shows that the SHAP values behave similarly as for the HH model with the only exception that the response to weather conditions is the complete opposite as expected according to our previous findings. It is also interesting to note how the weather conditions have higher significance in this model whilst the holiday feature ranks the lowest probably due to data imbalance in the limited random sampling.

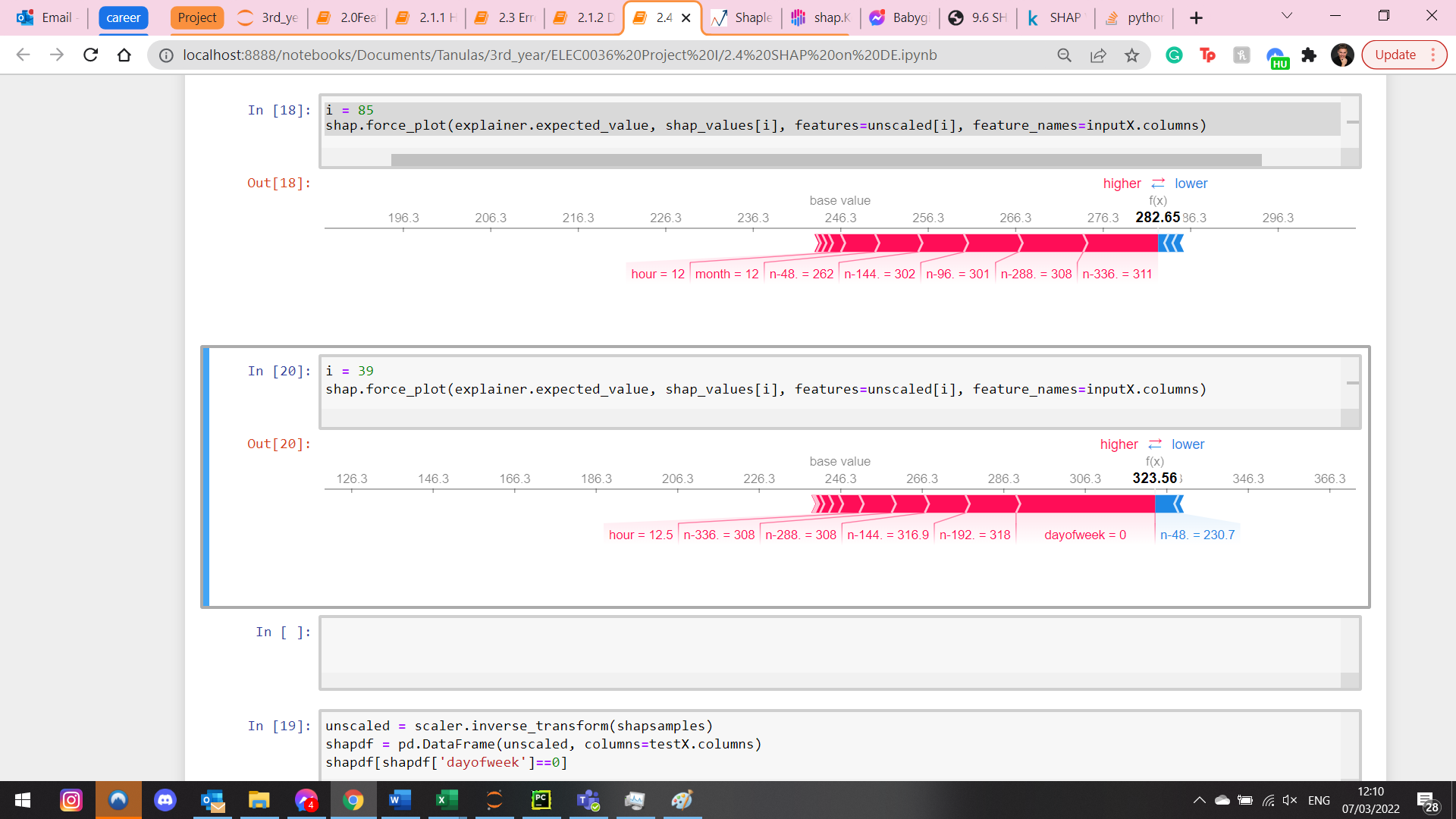


Figure 20 – local interpretation of accurate Monday 12:30

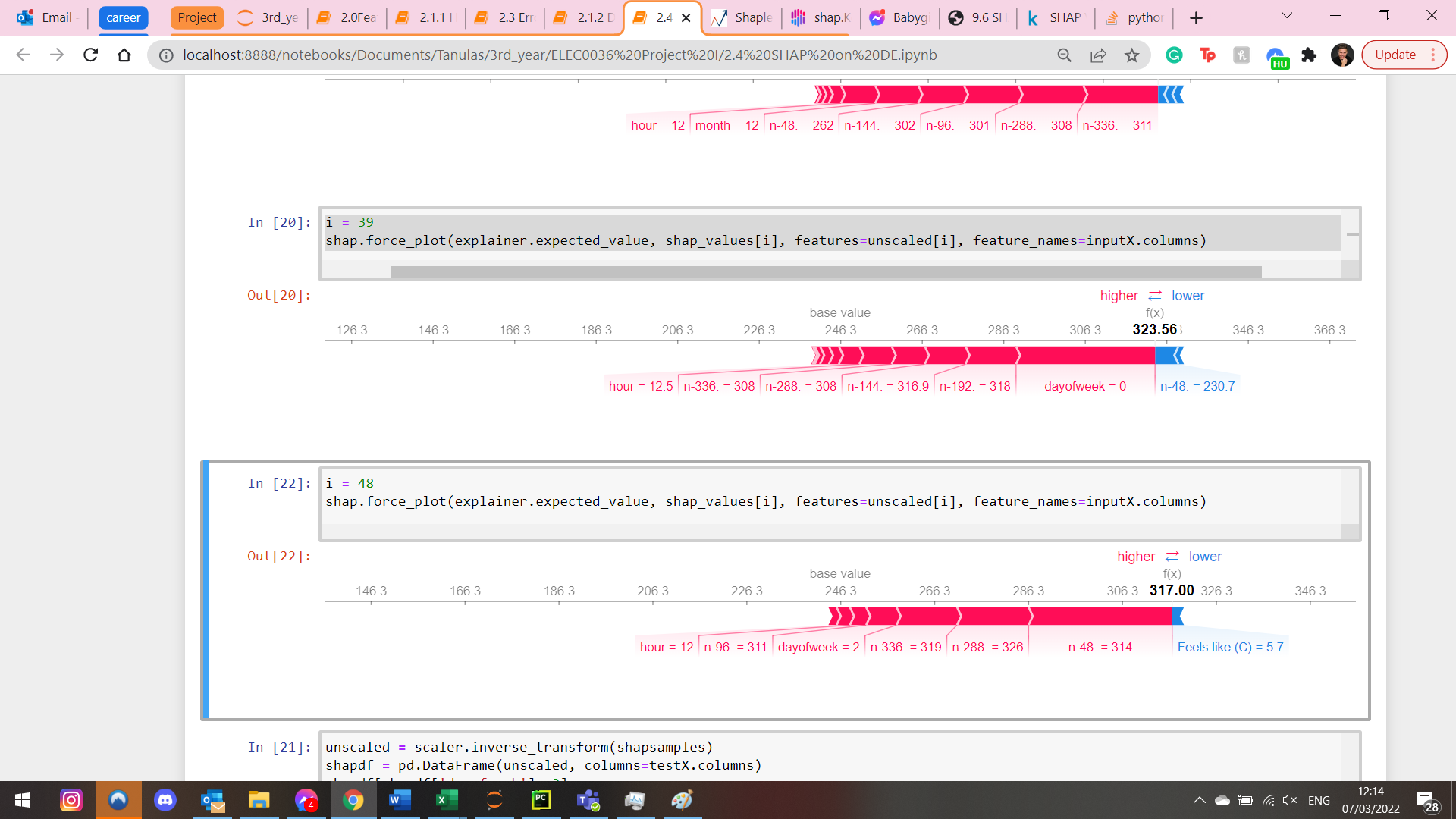


Figure 21 local interpretation of accurate Wednesday 12:00



Figure 22 local interpretation of accurate Saturday

Out of working hours 1:30 am



Figure 23 Local interpretation for point out of working hour 01:30

The local interpretations for DE SVM can be seen on figures 20-23. Whilst the feature effects are as expected there is an interesting difference when compared with HH interpretations. We can see that this model utilizes the weather conditions by adding more significance to them. It is also interesting to see how at one point the feels like feature had greater importance and how at another point the temperature had a more significant effect. This confirms the success of our feature engineering process and gives further reason to believe that not utilizing the weather conditions is something that needs to be targeted in future work.

### Summary

Overall, we could see that our SVM model for the DE block has very great performance on unseen data and that the design steps required to develop the model showed their results. It was interesting to see how on the other dataset the Random Forest model on HH block had significantly worse results despite following the exact same design steps. When performing further analysis of the model a possible weakness was identified that needs more focus in future research. We also found that the inaccuracies might be caused by random fluctuations that cannot be explained using the information in the dataset. The significant difference between the two datasets might de caused by the different nature of the building blocks. As the HH building is low-tech in its operation and is more dependant on occupant behaviour that is unpredictable. It introduces more randomness into the dataset that cannot be explained without an occupancy schedule dataset.

# Future work and conclusion

In this work we have shown the importance of building energy forecasting and how it could benefit society. We then revied the literature in the research area summarizing their main points and findings. It was pointed out that most works in the literature review have a poor or missing data pre processing stage. It was also criticized that most work was focused on one or only a few models whilst missing out on potential models that would fit their dataset more accurately. Another point of criticism was regarding the black box behaviour of their model with no effort to give an inn-depth analysis of the developed model.

After providing a detailed description of the relevant background theory this work showed the importance of the pre-processing step and described a sophisticated approach to clean the dataset from inconsistent points by restoring the original values that were distorted into outliers. Afterwards, the data enrichment step was demonstrated followed by extensive feature engineering efforts. Using the transformed dataset 4 machine learning models were developed. It has been shown that model performances can differ significantly depending on the dataset thus, highlighting the importance of exploring multiple models for machine learning projects. The performances of the developed model were similar to ones in the literature review especially when considering only studies with a similar level of residual in the dataset.

Lastly, the proposed models have been analysed extensively using visualization and interpretability techniques whilst showing the importance of this step to confirm the correctness of the inner workings of the model. We could also see how this step can identify possible points of weaknesses that can be targeted in future research.

Regarding the initial objectives of the work, we can conclude that except for a small error of 0,3% for the HH model we successfully met all objectives.

For future work could further investigate this issue that was presented in this paper. First, as for all research the work presented in the paper should be represented in future work to confirm the findings. It would be beneficial to test the models on a larger dataset with the dataset from years 2020-2022. When doing so one would need to introduce a lockdown feature to account for the unusual behaviour during covid times. Furthermore, given the poor performance of the RF model on the HH building block future work should focus on enhancing the performance on the HH model. A possible weakness for the HH model has been identified that needs to be further analysed and confirmed. A further potential performance issue for the HH model is identified as the dataset. Multiple instances were found where the behaviour of the target variable cannot be explained using the dataset. Acquiring an occupancy schedule might tackle this issue as it might offer explaining information about instances with strange behaviour such as a New Year’s Eve party at the end of 2019 in the DE building block that explains the never seen behaviour shown in the results section.

A further possibility of improvement is exploring more models. A promising method for developing more accurate models is using the hybrid approach as demonstrated by Hong, W.C. in [17]. It utilizes the strengths of different models and combines them into a hybrid model that has the joint strengths of the separate models. [18] Describes multiple variation of hybrid models and shows how each of them outperform both initial models when applied separately.

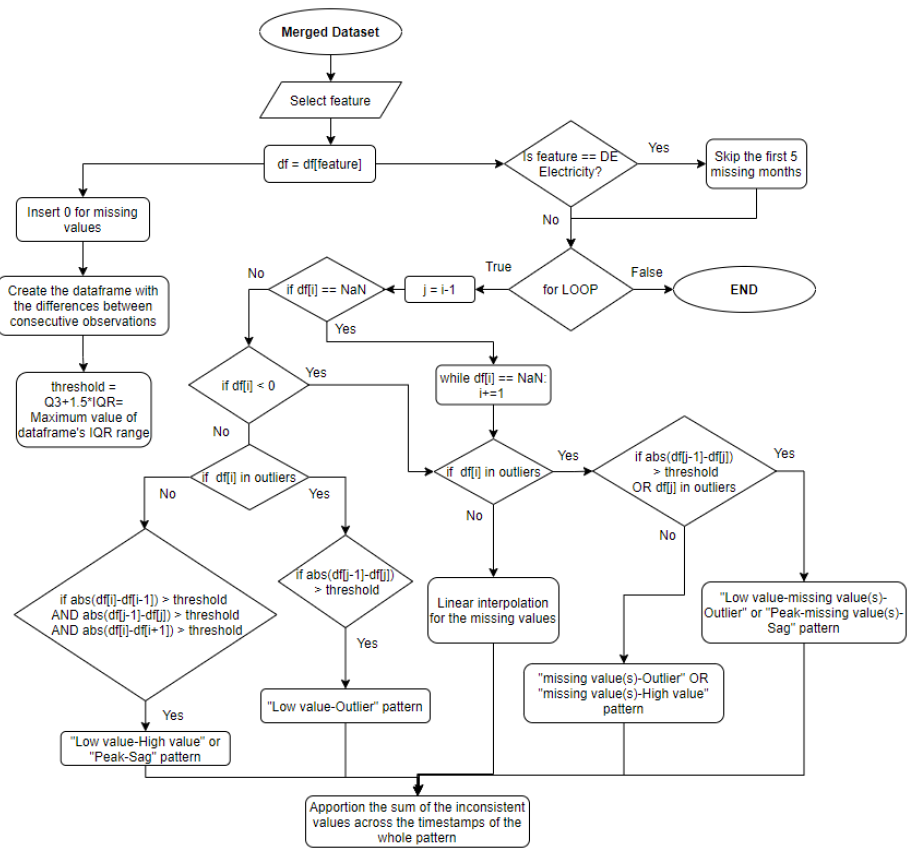
# References

|  |  |
| --- | --- |
| [1] | F. M. Hai-xiang Zhao, “A review on the prediction of building energy consumption,” *Renewable and Sustainable Energy Reviews,* vol. 16, no. 6, pp. 3586-3592, 2012. |
| [2] | I. &. C. A. &. C. S. &. C. T. Khan, “Fault Detection Analysis of Building Energy Consumption Using Data Mining Techniques,” *Energy Procedia,* vol. 42, pp. 557-566, 2013. |
| [3] | IEA, “World energy outlook 2011 executive summary,” International Energy Agency, 2011. |
| [4] | J. M. T. L. A. R. S. J. R. J. G. E. A. M. R. J. Alicia Troncoso Lora, “Time-Series Prediction: Application to the Short-Term electric energy demand,” in *Springer*, Berlin, Heidelberg, 2004. |
| [5] | J. A. F. A. S. Gabriel Happle, “A review on occupant behavior in urban building energy models,” *Energy and Buildings,* vol. 174, no. 0378-7788, pp. 276-292, 2018. |
| [6] | D.-S. T. Jui-Sheng Chou, “Forecasting energy consumption time series using machine learning techniques based on usage patterns of residential householders,” *Energy,* Vols. Volume 165, Part B, no. ISSN 0360-5442, pp. Pages 709-726, 2018. |
| [7] | I. M. Y. A. H. M. N. M. E. M. N Y Dahlan Rijalul Fahmi Mustapa, “Baseline energy modelling in an educational building campus for measurement and verification,” *International Conference on Electrical, Electronics and System Engineering,* pp. 67-72, 2017. |
| [8] | C. F. S. Y. Kaile Zhou, “Big data driven smart energy management: From big data to big insights,” *Renewable and Sustainable Energy Reviews,* vol. 56, no. 1364-0321, pp. 215-225, 2016. |
| [9] | P. Kalliga, “Analysis and optimization of building management systems through Data Analytics and Machine Learning,” unpublished, London, 2021. |
| [10] | A. Z. A.-G. S. J. Nizami, “Forecasting electric energy consumption using neural networks,” *Energy policy,* vol. 23, no. 12, pp. 1097-1104, 1995. |
| [11] | M. V. S.Karatasou, “Modeling and predicting building's energy use with artificial neural networks: Methods and results,” *Energy and buildings,* vol. 38, no. 8, pp. 949-958, 2006. |
| [12] | L. J.-h. L. L.-x. D. G. X. &. J. L. (. .. 2. ,. 1. 5.-5. Xuemei, “Building Cooling Load Forecasting Model Based on LS-SVM,” *2009 Asia-Pacific Conference on Information Processing,* vol. 1, pp. 55-58, 2009. |
| [13] | Z. L. H. Z. P. X. Yangyang Fu, “Using Support Vector Machine to Predict Next Day Electricity Load of Public Buildings with Sub-metering Devices,” *Procedia Engineering,* vol. 121, pp. 1016-1022, 2015. |
| [14] | Y. W. R. Z. R. S. S. S. A. Zeyu Wang, “Random Forest based hourly building energy prediction,” *Energy and Buildings,* vol. 171, pp. 11-25, 2018. |
| [15] | M. S. V. G. S. Karatasou, “Modeling and predicting building's energy use with artificial neural networks: Methods and results,” *Energy and Buildings,* vol. 38, no. 8, pp. 949-958, 2006. |
| [16] | L. Breiman, “Bagging predictors,” *Machine Learning,* vol. 24, pp. 123-140, 96. |
| [17] | W.-C. Hong, “Hybrid evolutionary algorithms in a SVR-based electric load forecasting model,” *International Journal of Electrical Power & Energy Systems,* vol. 31, no. 7-8, pp. 409-417, 2009. |
| [18] | F. Z. J. Y. S. E. L. K. W. S. Chirag Deb, “A review on time series forecasting technniques for building energy consumption,” *Renewable and Sustainable Energy Reviews,* vol. 74, pp. 902-924, 2017. |
| [19] | B. D. C. E. &. S. W. I. van Ruijven, “Amplification of future energy demand growth due to climate change,” *Nature Communications,* vol. 10, no. 2762, 2019. |
| [20] | D. Solyali, “A Comparative Analysis of Machine Learning Approaches for Short-/Long-Term Electricity Load Forecasting in Cyprus,” *Sustainability,* vol. 12, no. 9, p. 3612, 2020. |
| [21] | A. Z. A.-G. SSAK Javeed Nizami, “Forecasting electric energy consumption using neural networks,” *Energy Policy,* vol. 23, no. 12, pp. 1097-1104, 1995. |
| [22] | G. R. a. B. B. J. Newsham, “Building-Level Occupancy Data to Improve ARIMA-Based Electricity Use Forecasts,” in *Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*, New York, NY, USA, 2010. |
| [23] | L. J.-h. D. L. X. G. a. L. J. L. Xuemei, “Building Cooling Load Forecasting Model Based on LS-SVM,” in *2009 Asia-Pacific Conference on Information Processing*, vol. 1, Shenzhen, China, IEEE, 2009, pp. 55-58. |
| [24] | Z. L. H. Z. P. X. Yangyang Fu, “Using Support Vector Machine to Predict Next Day Electricity Load of Public Buildings with Sub-metering Devices,” *Procedia Engineering,* vol. 121, pp. 1016-1022, 2015. |
| [25] | Y. C. ,. X. S. ,. D. W. Junhua Zhuang, “Building Cooling Load Prediction Based on Time Series Method and Neural Networks,” *International Journal of Grid and Distributed Computing,* vol. 8, no. 4, pp. 105-114, 2015. |
| [26] | G. L. X. L. Y. W. Hongzhan Nie, “Hybrid of ARIMA and SVMs for Short-Term Load Forecasting,” *Energy Procedia,* vol. 16, no. C, pp. 1455-1460, 2012. |
| [27] | B. E. A. Y. M. R. Nasr G. E., “Neural networks in forecasting electrical energy consumption: univariate and multivariate approaches,” *INTERNATIONAL JOURNAL OF ENERGY RESEARCH,* vol. 26, no. 1, pp. 67-78, 2001. |

# Appendices

Appendix A

This appendix describes the process of recreating the flowchart from [9]. The flowchart has been copied here for comfort:



Whilst this report is mainly interested in the electrical data the work of Kalliga, Polyxeni was using the other features as well.

The outlier patterns of other features were the following:

Heating:

* Peak – missing value(s) – sag
* Missing value(s) – High value

Cooling

* Peak – sag

Water capacity

* Missing value(s) – outlier

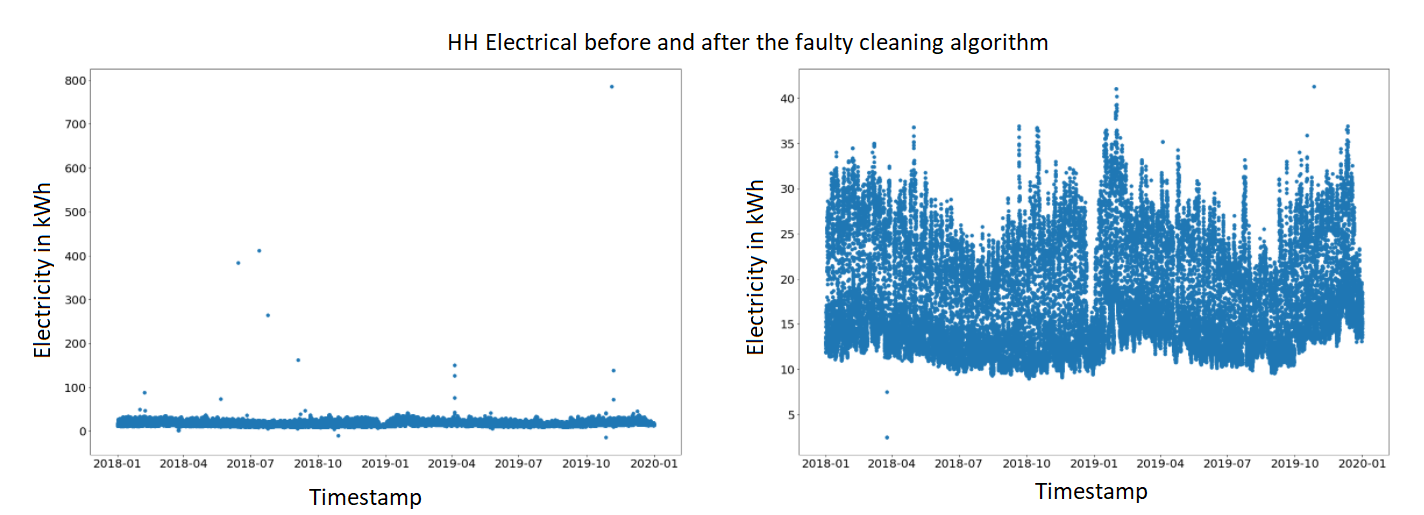


Figure A1 - Output of Flowchart code without modification for HH Electrical

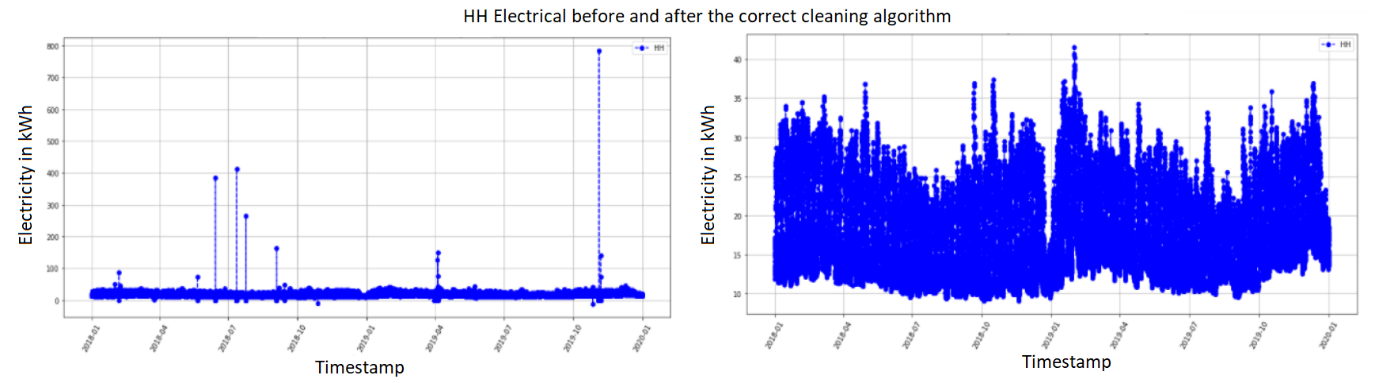


Figure A2 - output of cleaning algorithm taken from [9]

Whilst the cleaning algorithm show improvement there are still differences to the results in comparison to When analysing differing points between Figure 7 and 8 one can replay the flowchart algorithm by hand. When analysing the output, the results suggested that the flowchart describing the cleaning algorithm was incorrect. The outputs for other features were even worse, strengthening this assumption. In a meeting with Polyxeni, Kalliga this was confirmed.

The issue could be solved easily by adding parts to the code that were missing from the flowchart.

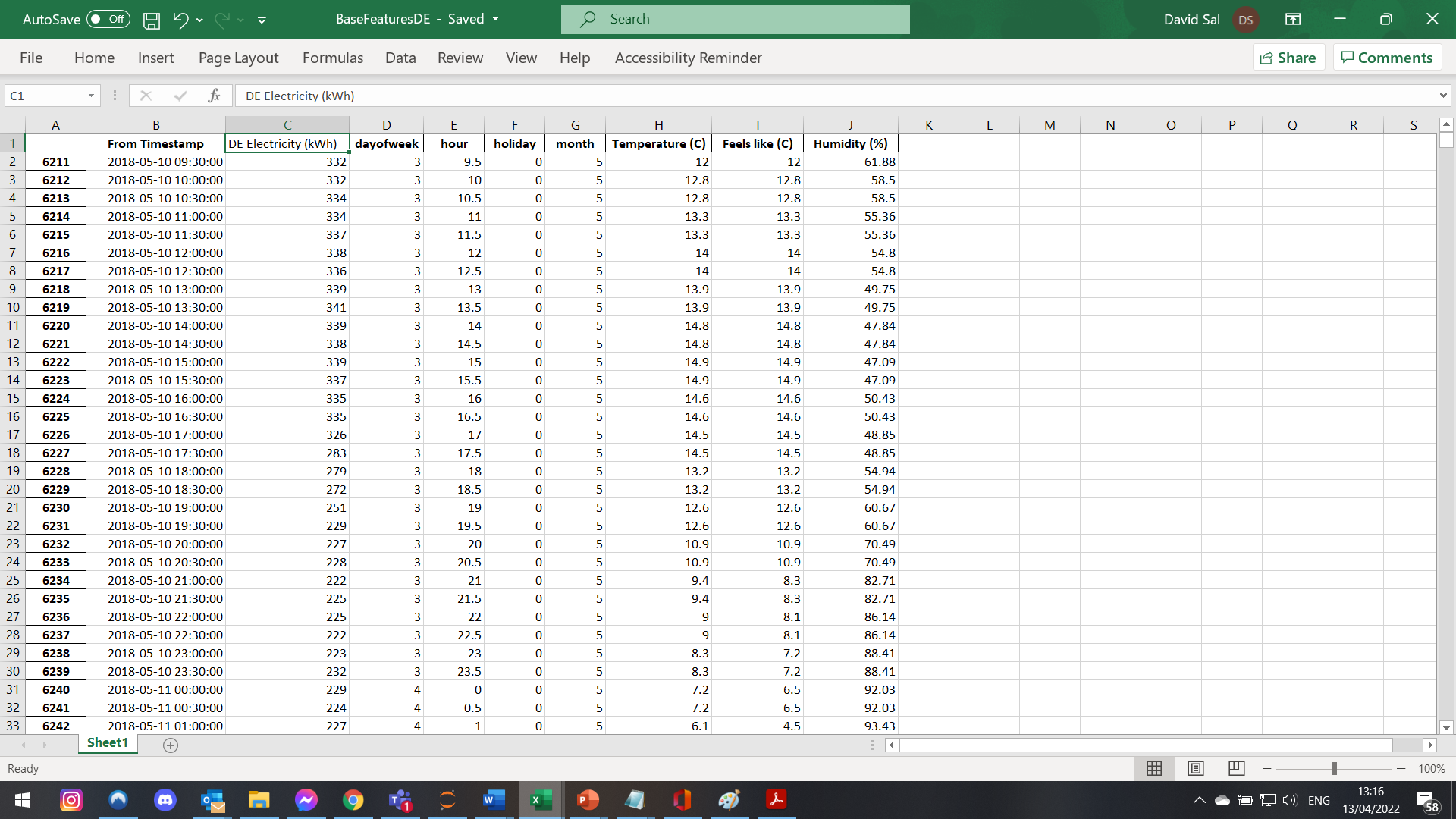
The changes were the following:

* Cleaning sag peak patterns in feature 2
* Cleaning sag – zeros – peak in feature 3
* Cleaning sags in feature 2 through linear interpolation
* Cleaning feature 6 where we have zeros – outlier that should be no values – outliers but it has been manually adjusted in the original dataset.

After adding the parts described above the code properly cleaned the dataset and the results agreed with the ones presented in the previous student’s paper.

Appendix B

The following code expects a dataset of the following structure:



|  |
| --- |
| df = pd.read\_excel("BaseFeaturesDE.xlsx")  df = df.drop('Unnamed: 0', axis = 1)  lagNumberStart = 1  lagNumberEnd = 8  for i in range(lagNumberStart, lagNumberEnd):  name = "n-{}.".format(48\*i)  df[name] = df['DE Electricity (kWh)'].shift(48\*i)  i = list(range(lagNumberEnd\*48))  df = df.drop(i)  feature\_cols = ['dayofweek', 'hour', 'month', 'holiday', 'Temperature (C)', 'Feels like (C)', 'Humidity (%)']  for i in range(lagNumberStart, lagNumberEnd):  name = "n-{}.".format(48\*i)  feature\_cols.append(name)    X = df[feature\_cols].copy() #input features for fitting  size = -int(len(df)\*0.20)  Xinput = df.iloc[:size]  Xtest = df.iloc[size:] # 20%  size2 = -int(len(Xinput)\*0.25)  Xtrain = Xinput.iloc[:size2].copy() #60%  Xvalidate = Xinput.iloc[size2:].copy() #20%  trainX = Xtrain[feature\_cols] # Features  trainy = Xtrain['DE Electricity (kWh)'] # Target variable  validateX = Xvalidate[feature\_cols] # Features  validatey = Xvalidate['DE Electricity (kWh)'] # Target variable  testX = Xtest[feature\_cols] # Features  testy = Xtest['DE Electricity (kWh)'] # Target variable  inputX = Xinput[feature\_cols] # Features  inputy = Xinput['DE Electricity (kWh)'] # Target variable  scaler = StandardScaler() #scaling  scaler.fit(X)  trainXscaled = scaler.transform(trainX)  validateXscaled = scaler.transform(validateX)  testXscaled = scaler.transform(testX)  inputXscaled = scaler.transform(inputX) |