Technische Universiteit Delft

Energy Study of Drying

Using Machine Learning To Predict The Energy Consumption Of An Industrial Powder Drying Process

Mohammed el Ouasgiri

TU Delft

Energy Study of Drying

Using Machine Learning To Predict The Energy Consumption Of An Industrial Powder Drying Process

by

Mohammed el Ouasgiri

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday September 8, 2022 at 12:00 PM.

Student number:	4499166	
Project duration:	August 16, 2021 – March 16, 2022	
Thesis committee:	Dr. P. Chen,	TU Delft, supervisor
	Prof. dr. ir. A. Papapantoleon,	TU Delft, chair
	R. Verhoek,	Abbott, supervisor

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft Abbott

Abstract

In this thesis, we use data science / statistical techniques to better understand the energy consumption behind a powder drying facility located in Zwolle, as part of Abbott's initiative to better manage its energy consumption. As powder drying is by far the facility's most energy intensive process, this project therefore focuses exclusively on powder drying.
The primary goal is to develop an accurate predictive model for the energy consumption, which can be used as a baseline to assess whenever or wherever the largest changes in energy efficiency took place. This can in turn be used to influence the energy policy, and may for example be used to assess what kind of retrofits can have the largest positive effects on efficiency.
Hereby we consider a variety of predictive models in increasing order of complexity. We also make our own contribution by describing a predictive model based on cluster-analysis, where we fit separate models on each cluster in order to better capture their specific patterns. It turns out that the latter approach is capable of drastically improving predictive performance.

Preface

This thesis is the result of several months of hard work conducted at the Zwolle production facility of the international pharmaceutical company Abbott, as part of a graduation project in the Master's degree of Applied Mathematics at TU Delft.

I can not overstate the challenges that presented itself while finishing this project, in no small part due to the global ramifications that the Covid virus caused. Regardless, despite the enormous challenges that came with primarily working from home, I would like to present my gratitude to Robin Verhoek, Ronald Terlouw and Alfons Rijpkema, who were always willing to take time out of their busy schedules to address any questions I had. Furthermore, I extend my gratitude to Piao Chen, who was always keenly interested in my progress and gave me helpful advice along the way, and also to Antonis Papapantoleon for his willingness to serve as the committee chair.
This therefore marks the end of a long fruitful period at TU Delft, and hopefully leads to better things to come.

Mohammed el Ouasgiri
Delft, August 2022

Terms and Definitions

Energy

The terms and definitions below are adopted from ISO 50006-2014 [3].

baseline period
Defined period of time used to compare energy performance with the reporting period

energy
Electricity, fuels, steam, heat, compressed air, and other like media

energy baseline (EnB)
Quantitative reference(s) providing a basis for comparison of energy performance
Note 1 to entry: An energy baseline reflects a specified period of time.
Note 2 to entry: The energy baseline is also used for calculation of energy savings, as a reference before and after implementation of energy performance improvement actions.

energy performance
Measurable results related to energy efficiency, energy use and energy consumption

energy performance indicator (EnPI)
Quantitative value or measure of energy performance
Note 1 to entry: EnPIs could be expressed as a simple metric, ratio or a more complex model.

energy review
Analysis of energy efficiency, energy use and energy consumption based on data and other information, leading to identification of SEUs and opportunities for energy performance improvement

relevant variable
Quantifiable factor that impacts energy performance and routinely changes

reporting period
Defined period of time selected for calculation and reporting of energy performance
EXAMPLE: The period for which an organization wants to assess changes in EnPIs relative to the EnB period.

significant energy use (SEU)
Energy use accounting for substantial energy consumption and/or offering considerable potential for energy performance improvement

specific energy consumption (SEC)
Energy consumption required per unit of product

static factor
Identified factor that impacts energy performance and does not routinely change

Drying process

We only list the main terms here, for more specific terms we refer to chapter 2.

FP-tank
Finished product tank (product that is received prior to drying)

Hipex
Sub-system that pre-heats the product prior to evaporation

Filter
Removes contaminants from the air or product

Buffer tanks
Balance tanks that store product between drying stages (to ensure constant product flow).

Thick milk
Concentrate, the product after it passes the evaporation stage

UHT
Ultra High Temperature, refers to heating stage between evaporation and drying to improve microbiological quality

DMB
"Dikke Melk Bak", serves as buffer tank prior to drying stage

Nozzle
Atomizer, i.e. a spraying mechanism that serves to pulverize the product into small droplets

Cyclone
Conic shaped installation that separates residue powder particles from the exhaust air

Bag filter
Filters that separate finer particles from the exhaust air (which were not separated in cyclones)

Effects
Calendria ("trappen"), refers to the multiple stages in evaporation or drying

Manifold
A network of pipes, through which the product flow occurs

Reimeltsystem
Refers collectively to the transportsystems and storage silos for the powder after drying

Fines return
Powder particles that are retrieved from cyclones/bag filters.

CIP
Cleaning in Progress, used to indicate maintenance periods. When the CIP state is idle, there is no maintenance taking place.

Energy carrier
Specific medium that supplies / contains the energy.
EXAMPLES: Gas, steam, electricity, compressed air.

Nomenclature

These definitions apply unless otherwise specified.

p	Amount of variables	N	Amount of observations	$\boldsymbol{X}$	Design matrix containing input variables
X_j, x_j	j-th input variable	$\boldsymbol{y}$	Actual output values	$\hat{\boldsymbol{y}}$	Predicted output values
Y, Y_j	Partially observed variables	p_1	Amount of variables in Y	X	Fully observed variables
p_2	Amount of variables in X	Y_j^{obs}	Observed part of Y_j	Y_j^{mis}	Unobserved part of Y_j
$\dot{Y}$	Imputed variant of Y	$\{\cdot\}_{-j}$	All indices aside from j	l	Amount of iterations in MICE
m	Amount of imputed datasets in MICE	X_t	Time-series	$d(\cdot,\cdot)$	Distance value
$C_{(\cdot)}$	A specific cluster	K	Amount of clusters	α	Significance level for prediction intervals, tuning parameter
$(\cdot)_i$	i-th quantile, for i between 0 and 1	IQR	Inter-quantile range	$D_{(\cdot)}$	Cook's distance
$GVIF, VIF$	(Generalized) Variance Inflation Factor	R^2	Squared R	df	Degrees of freedom
$RMSE$	Root Mean Squared Error	MAE	Mean Absolute Error	PI	Prediction interval
$L_{(\cdot)}$	Lower boundary of PI	$U_{(\cdot)}$	Upper boundary of PI	λ	Penalization parameters for regularized models
ϵ	Noise, tuning parameter for DBSCAN	$\boldsymbol{\beta}$	Regression coefficients	RSS	Residual sum of squares
$\hat{\boldsymbol{y}}_N, \hat{\mu}_N$	Regression function	$\hat{\boldsymbol{\rho}}_N$	Regression function for absolute residuals	η	Bias
η_N	Model variance noise	B	Amount of bootstrap replications	$\hat{\gamma}$	No-information error rate
$\hat{R}$	Relative overfitting rate	$(\cdot)_{perc}$	All percentiles between the first and 99th	$L(\cdot,\cdot)$	Loss function
$\hat{y}_\tau$	Quantile regression function for quantile τ	OLS	Ordinary Least Squares	RF	Random Forests
$LM_{(\cdot)}$	Variants of the OLS model	$S-CI$	Split conformal inference	t_j	t-value
$\boldsymbol{H}$	"Hat" matrix	d_j, δ_j	j-th singular value of matrix	T	Score matrix
$Z_{(\cdot)}$	Derived/Score variable	M	Amount of derived variables	P	Loadings matrix
$\phi_{(\cdot)}$	Loadings vector	PCR	Principal Component Regression	PLS	Partial Least Squares
PC	Principal Component	λ_j	j-th eigenvalue	v_j	j-th eigenvector
CV	Cross-validation	$GCV(\lambda)$	Generalized cross-validation criterium	$\mathbb{C}$	Collection of reflected pairs for MARS
$\mathbb{M}$	Set of functions used by MARS	$h_{(\cdot)}$	Basis function used by MARS	$R_{(\cdot)}, R_{l(x,\theta)}$	Regions/subspaces used by RF
T	Trees in RF	$C_\alpha(T)$	Complexity parameter from RF	$\boldsymbol{Z}^*$	Bootstrapped sample
OOB	Out-of-bag	QRF	Quantile Regression Forests	$A_{(\cdot)}$	Subsets used by S-CI
$MinPnts$	Tuning parameter for DBSCAN	CVI	Cluster Validation Index	$\mathbb{S}$	Cluster compactness for S_{Dbw}
$\mathbb{V}$	Vector of variances for set of variables	$\gamma_{kk'}(\cdot)$	Relative density of point between two clusters C_k and $C_{k'}$	$\mathbb{G}$	Cluster separability for S_{Dbw}
m_k	Cluster centroid for cluster C_k	$N_\epsilon(\cdot)$	Sphere of radius ϵ as used by DBSCAN	$Q_\alpha(\cdot)$	α-th quantile

Contents

1

Introduction

This thesis concerns itself with predicting, and analysing the energy consumption of an industrial dryer located in Abbott Zwolle. As a significant energy user, industrial drying is a process that requires a huge amount of energy, and is therefore a prime target for energy management programs within companies such as Abbott to improve energy efficiency and/or decrease energy consumption. More specifically, it has been estimated that industrial drying accounts for 10-25% of energy use in countries such as Canada, USA, France, UK, Denmark and Germany[13]. In the branch located in Zwolle, drying makes up for 50% of total energy use.

The practical consequences of improving energy aspects, especially in energy intensive industrial processes, should not be understated. Amidst growing concerns of climate change, depletion of finite resources such as fossil fuels and other environmental concerns, it should become a high priority for companies to manage their energy consumption and avoid needless waste. Of course, this provides benefits to companies as well, through lower energy costs and positive public reception for example.

Industrial drying is a prime target towards this end, not only because of its high energy usage as previously stated, but also because of the low energy efficiency that dryers tend to have [13] [7]. It is primarily through these two conditions, that global energy management institutions like ISO, EVO and CIPEC recommend to critically assess processes that satisfy these conditions [3], [2], [1]. Hereby, an organization is obliged to define Energy Performance Indicators (EnPIs) for each significant energy user that:

- are appropriate for measuring and monitoring its energy performance;
- enable the organization to demonstrate energy performance changes.

Once calculated, the EnPI values are compared with an energy baseline to indicate whether energy performance has improved or worsened.
Additionally, industrial drying is a cornerstone of our modern society, as it allows the production and distribution of many vital goods throughout the world. A good example is powder drying, which allows otherwise perishable goods to be maintained, and also provides economical benefits through lower transport costs (as dried products weigh less). Therefore, the energy aspect of industrial drying is further exacerbated by the global dependence on it.

Because of the reasons outlined above, a great deal of literature has been dedicated towards studying the energy consumption of drying processes. For some examples of related case-studies in food/powder drying, see [13], [50], [16] and [14]. While it is not precisely the same as the powder producing process that we will consider, the corresponding literature discusses many relevant points. Historically, research studies tend to model the problem from two perspectives.
The first approach is the "theoretical" approach, which was traditionally used before the advent of data-centric approaches like machine learning. In this case, one tries to define a theoretical model for the underlying drying process, mainly through the use of physical equations. While it has been repeatedly shown that such models can provide some insight into these processes, some serious complications arise from this approach.
Firstly, it takes a great understanding of the process to define and understand a theoretical model, which makes this approach inaccessible to most people. Secondly, industrial drying is a notoriously difficult problem for modelling, as there are generally many interlinked sub-processes involved and other non-linearities [7], [40], [52]. It therefore requires great effort and time to model one specific process. As there are generally many possible drying processes due to combinations of fixed variables (type of dryer, type of product, etc.), this means that a separate model would need to be developed for every new problem. Accordingly, the prediction ability of theoretical models, especially in real-time applications, is very limited [52].

An important consequence, is that this theoretical approach is unsuitable for control scenarios, where one needs to quickly forecast and control critical variables when the underlying conditions change.

The second approach centers around the data, i.e. machine learning (ML). Due to increasing computational capabilities, this approach has become very popular in recent years. Especially in drying literature, it has outpaced the theoretical approach due to the fact that it can circumvent the previously mentioned problems [52], [7]. Namely, as long as enough data is available and an adequate rule-discovery algorithm is used, a computer could learn all possible physical relationships. This also removes the barrier of entry caused by the use of complex physical models, therefore making the ML-based approach accessible to more researchers. Additionally, the ML-based models are based on real observations, therefore they could be better capable of describing the true conditions of a process.
For real-time adaptation, the ML-based approach is therefore especially well-suited to quickly assess the impact of various actions, and therefore ML can turn data into actionable insights. Within Abbott, this might mean that one needs to understand how energy consumption changes, when a completely new product type enters the drying process.

Overall, ML models try to optimize a certain performance criterion. It does this by iteratively tuning the model parameters through a representative training data set. The quality of data, the statistical algorithm responsible for optimization and the specific form of the performance criterion are crucial.
Through combinations of the latter two, many kinds of ML models have been developed over the years [36]. Traditionally, we have the classical approaches like linear regression and time series analysis [15], which restrict themselves to optimizing a fixed functional relationship between output and input. While this offers good interpretability, this also makes them inflexible and less capable of representing complex non-linear processes like industrial drying.
More recently, we have seen the emergence of more advanced ML models that do not strictly adhere to a specific form [36], [39]. Models like Artificial Neural Networks (ANN) use elaborate constructs, that can in essence represent any kind of complex relationship, at the cost of interpretability. Therefore, the main means within the drying research community to model drying processes, has shifted towards ML models (for a general overview see [52], [40], [7], [57] and for some specific applications see [16], [50], [13]).

The main goals that we wish to achieve in this thesis, relates to finding an accurate predictive ML model for the drying process within Abbott Zwolle such that its energy consumption can be predicted from a curated list of input variables, and to demonstrate how such a model could be used to find the largest changes in energy efficiency. The main novelty in this regard, comes from the development and comparison of a selected number of diverse techniques. Whereas in related areas such as the energy consumption of buildings and national power grids, the range of used methods is large [29], [57], [32], it appears that within the drying literature we primarily see the use of ANN only [7] [13] [40] [52]. Therefore, we would like to study other ML methods and assess the relative (dis)advantages.
Based on the predictive accuracy, we will select the most accurate ML model to be used as an EnPI and use it to answer the following research questions:

1: Can real-time models be developed to measure energy performance, such that it can be verified when the powder drying process shows the largest changes in efficiency?
1.1: Which model offers the best predictive accuracy, both in terms of point and interval accuracy?
1.1.1: What is the accuracy for periods of active production?
1.1.2: What is the accuracy in transitory periods between products (tail-ends of each production-cycle)?
1.1.3: How well does the considered model generalize to completely new products?
1.2: What is a general method of using the predictive model to measure changes in energy efficiency?

2: What are important dependencies in the energy requirements?
2.1: How can the dependencies be visualized?
2.2: Are there common elements in important variables between the most accurate models?

Research question 1.1 is aimed at finding the most accurate predictive model, while question 1.2 is aimed at demonstrating how this model could be used to measure the changes in efficiency. Aside from the overall situation, the predictive accuracy will also be separately assessed for the top 10 products, for periods of active production and for transition periods. The models that we shall consider are both traditional methods as well as more advanced ML models. Eventually, the most accurate model will be used to demonstrate how the largest changes in energy efficiency could be found. This is done by following the procedure as outlined in [3], and as we have briefly described before. Namely, we use the predicted values from the model as an energy baseline, as they represent the circumstances from a previous period that we want to compare against. Energy efficiency can then be assessed by comparing the recent, actual values against the baseline, and an increase/decrease in efficiency is indicated by the former being smaller/larger.
For research question 2 we will use visualizations and model-specific variable importance measures to better understand

how the dependencies look like, where we will generally subdivide the variables into 5 groups representing distinct parts of the drying process: Evaporators, UHTs, tower, dehumidifier and fluidizer. Eventually, to find the most important variables, we select those variables that consistently appear as an important variable for the most accurate models.

The structure of the report is as follows. In chapter 2 we give the necessary background behind the powder drying process, where we give a general description of the whole process. After that we start with the data exploration in chapter 3, and in chapters 4 and 5 we will study the data in more detail by considering various predictive models, where we respectively consider linear and non-linear models. Finally, in chapter 6 we summarize our findings and give some recommendations.

2

Drying process

In this chapter, we will give an explanation of the entire drying process that takes place in the production plant located in Zwolle. During this process, nutritional powders are produced that are targeted towards susceptible demographics throughout the world. Therefore the quality requirements of the products are very strict, and accordingly a very elaborate drying process was devised to adhere to these requirements. We will first give a brief run-down of the entire process, before focusing on each part in the drying department.

Before the product arrives at the drying department, the base components are weighed/measured and then processed at the Powder Processing department. The processing department is mainly responsible for mixing, filtering, homogenizing, pasteurizing and cooling the initial base ingredients into a liquid product, which is then stored in one of four finished product tanks (FP-tanks). Finally, once the product passes some tests, it is cleared for the drying process.

From the FP-tanks, the product is pumped to the so-called Hipex. The Hipex is a heat-exchanger, which heats and pasteurizes the product to ensure it has the right temperature for the subsequent evaporation, and also to improve its microbiological quality.
The product then moves to the evaporators, where further heating takes place in order to evaporate water from the product. The product now becomes thicker and has a higher density. After passing through a filter to remove contaminants, the product is stored in a balance tank and at this stage is commonly referred to as "dikke melk" (*thick milk*). The product is now moved to the "Ultra Hoge Temperatuur sterilisatie" (UHT, *Ultra High Temperature Sterilization*).

In the UHT, the product is very briefly (a few seconds) heated to a very high temperature (in the range of 105 to 120 °C, depending on the type of product). This part of the drying process mainly serves to kill off micro-organisms, without causing too much cook taste. After passing through another filter, the thick milk is stored in a "dikke melk bak" (DMB), which is a buffer tank for the final part in the tower.

From the DMB, the thick milk is pumped with a high pressure pump to the top of the tower. From there, the liquid product is sprayed into the inside of the tower through a set of nozzles, while very hot air is inserted. Through the mixing of the liquid and the air, the product pulverizes into fine powder particles, before exiting the tower at the bottom where a vibrating fluid bed dryer/fluidizer is located. The air that was used for heating will also contain powder particles, so through separate installations ("cyclonen" and bag filters) the air and powder are separated.
The powder now enters its final drying stage, where it is dried and cooled with cool air. Finally, the product is transported through air to the storage silos, where it is ready to be packaged and transported.
A schematic overview of the drying process is shown in figure 2.1.

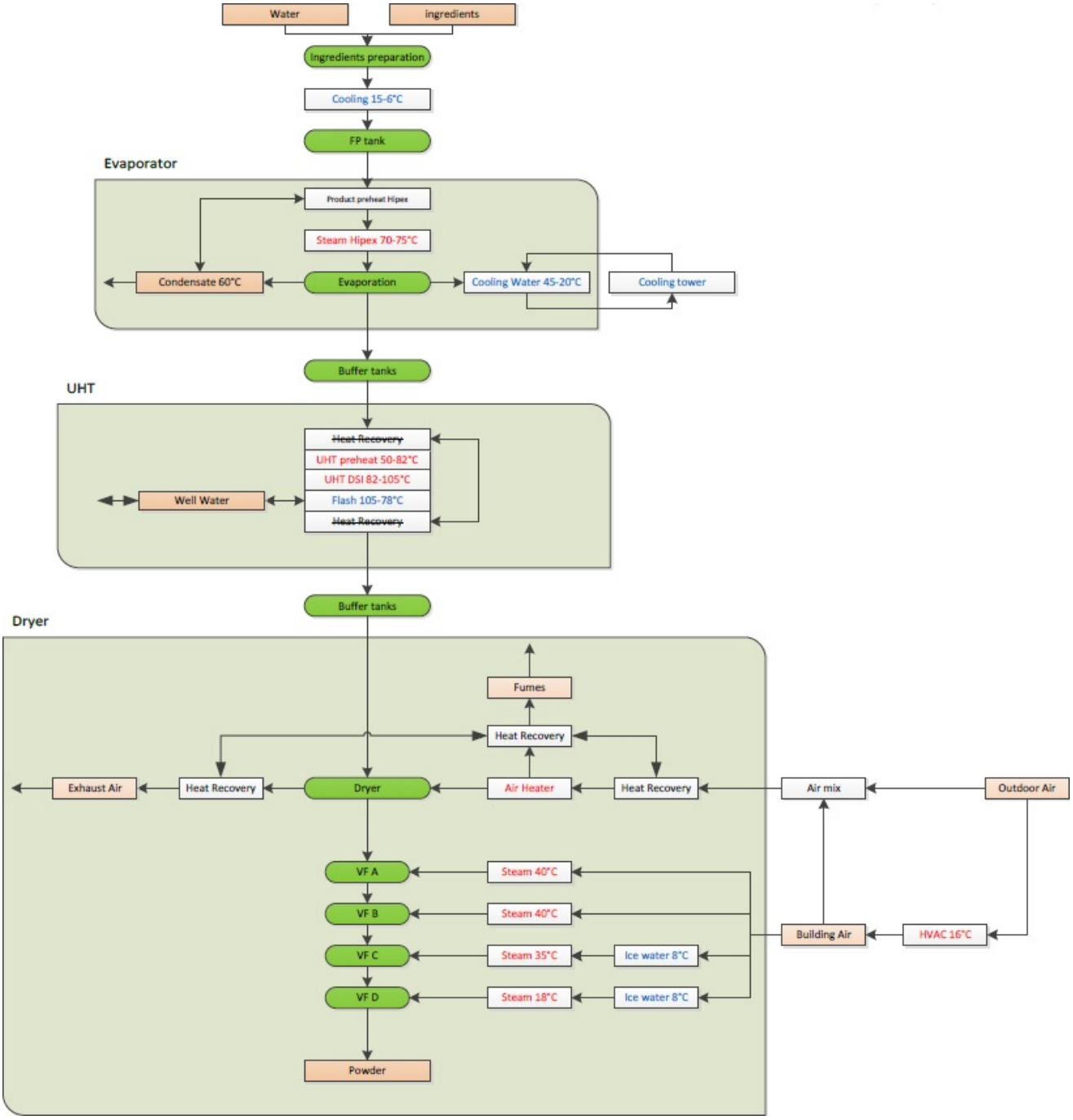


Figure 2.1: A schematic overview of the drying process. Orange nodes indicate inputs/outputs, green nodes indicate the process steps and blue/red indicate the cooling/heating steps.

We now give a more detailed description of each part in the drying process.

2.1. Evaporator

The specific evaporator that is employed, is the falling film evaporator as depicted in figure 2.2. As the product arrives at the top of the evaporator, it falls down through the different downpipes, while it is indirectly heated by steam through the boundaries of the pipes (see figure 2.3). This occurs over two stages ("trappen"), eventually causing the evaporation of water from the product.
There is a certain physical phenomenon that is exploited to decrease the energy costs associated with this process, as the evaporation of water is normally energy-intensive. This phenomenon has to do with maintaining a vacuum (zero absolute pressure) inside the pipes, as the boiling temperature decreases in low pressure environments. Therefore, the temperature difference between the product and heating medium increases, causing a bigger exchange of heat and thus increasing the capacity of the evaporator. Energy savings are also achieved by the usage of lower temperature steam, which means that the released vapour during the first stage can be used for the heating during the second stage. Aside from energy savings, decreasing the boiling temperature means that the product suffers less quality loss.

In the following, we list every component of the evaporator, alongside some additional information whenever important.

• **Withdrawal tank** ("intrekbak"): Receives the product from the FP tanks, and serves as a buffer tank. This ensures that there is a constant product flow to the evaporator.

• **Hipex**: pre-heater and pasteurizer. Ensures that the product has the right temperature before evaporation, and also improves microbiological quality of product.

• **Distributor** ("verdeler"): Distributes the product evenly across the downpipes, and also makes sure the product flows down along the downpipes instead of simply falling down. If the amount of product varies too much from pipe to pipe, it is easier for burning of the product to occur. For the process it is also important that the right amount of product flows down: when there is too little the pipe will silt up (i.e. the flow is blocked). When there is too much product, there will be too little heating.

• **Heater body** ("verhitterlichaam"): As the product flows down through the downpipes, it is indirectly heated from the outside by steam. This ensures evaporation of water, and consequently the product becomes more dense. How well heat exchange occurs depends on many factors, but some important ones include speed, viscosity and temperature of the product. Therefore, to ensure good heat exhange (and thus lower energy consumption), it is vital to study these factors.
The vapour that releases from the product is commonly referred to as "brüdendamp", and plays an important role. For example, to maintain low pressure inside the pipe (needed for decreasing boiling temperature), the brüdendamp is condensed in the condensor. This decreases pressure, because liquid requires a much smaller volume than gas.
The evaporation takes place over two stages, after which the product is collected at the bottom as concentrate ("concentraat").

• **Steam separator** ("damp-afscheider"): Receives the brüdendamp, and separates the liquid product drops from the vapour. Therefore product is saved, and the brüdendamp can be used for other purposes.

• **Thermocompressor**: Receives the brüdendamp from the steam separator, and heats it so that it can be used as heating medium for evaporation.

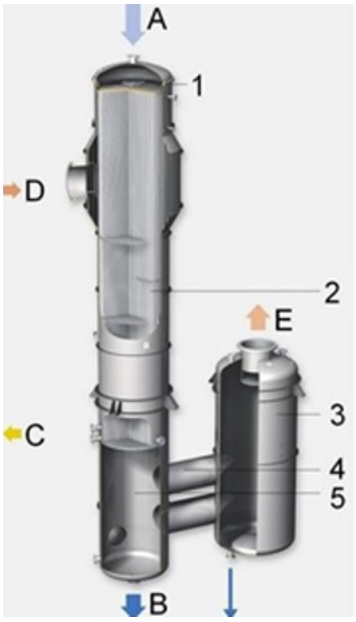
• **Condensor**: Condenses the brüdendamp from the second evaporation stage, to maintain vacuum inside the evaporator.

• **Centrifugal pumps**: Transport the batch from the FP-tanks to the evaporator, through a system of valves called manifold 204.

• **Vacuum pumps**: Removes non-condensable gasses from brüdendamp. Also is used to attain vacuum during beginning of production process, as no condensate has yet been produced.

• **Water tank**: Stores the condensed water from the condensor, which is not used for heating. The water is disposed as sewage.

The concentrate that is produced, or "dikke melk", passes through one more filter before it is transported through manifold 204 to the buffertank for UHT.



A= producttoevoer
B = productafvoer
C = condensaat
D = opgewarmde brüdendamp
E = brüdendamp

1 = verdeler
2 = verhitterlichaam
3 = dampafscheider
4 = toevoer dampafscheider
5 = opvang concentraat

Figure 2.2: Falling film evaporator.

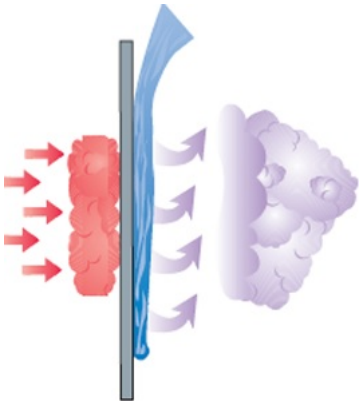


Figure 2.3: Heat exchange to evaporate water, using steam.

2.2. UHT

From a buffertank, the product is pumped to the UHT through a centrifugal and lobe pump. The latter pump is required for transporting viscous products. Like the Hipex, the UHT is a heat exchanger, see figure 2.4. However, in this case the temperature is increased even more for proper sterilization.
The UHT first consists of two sections, in which we have multiple small tubes inside a bigger, single tube. While the product flows through the smaller tubes, it is heated by hot water that surrounds the inner tubes. Afterwards, the product reaches the direct steam injector (DSI), where it is directly injected with steam. This causes rapid heating, which lasts for a short period to kill off micro-organisms and prevent too much cook taste.
Because of the injection, the product now contains a little bit of water which needs to be removed. This occurs in the flash barrel ("flash vat"). It exploits the same phenomenon as inside the evaporator, as in that it utilizes low pressure to decrease the boiling temperature. Therefore, the product immediately start boiling as it enters, and water is likewise removed through evaporation. The UHT therefore also contains a vacuum pump, for the same reason as the evaporator.
The product now passes another filter which removes all undesirable elements, and a centrifugal pump transports the product through manifold 204 to the buffer tank for the drying tower ("dikke melk bak").

2.3. Drying Tower

The last part of the drying process, which is also the most energy intensive, takes place in the tower and its fluidizer / bed dryer. The tower itself is responsible for turning the product into a powder, and the bed dryer that lies underneath it makes sure the product is properly dried and cooled before it is sent to the packaging department. We now describe both

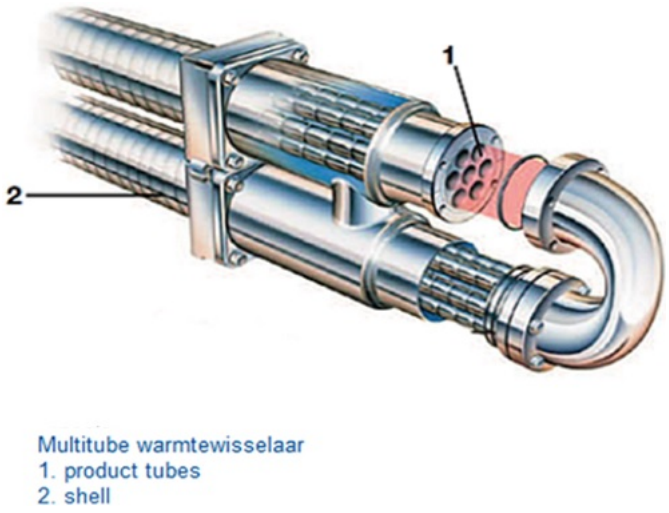


Figure 2.4: Schematic overview of first two sections in UHT.

the product- and airline, which are schematically shown in figure 2.5.

2.3.1. Product line

From the DMB, the product is pumped through manifold 205 to a high pressure pump, which in turn pumps the product to the lances on top of the tower. On the way, it may be possible for nitrogen to be injected, depending on the product type.
Each lance has a mounted nozzle, which makes sure that the product is sprayed as small droplets around the inside of the tower. At the same time, hot air is injected to evaporate the product. This process takes place so quickly, that the product immediately pulverizes upon exiting the nozzle, and then falls down into the dryer.

Aspects of the pulverization can be controlled by the nozzle, both through its composition as well as the angle that the product makes after exit (which in turn also depends on the pressure and properties of the product). The spraying pattern has an impact on how much heating takes place, and therefore on the energy consumption. Aside from this, the quality of the product (for example, solubility) depends on it.

The dryer that lies underneath the tower is a vibrating fluidized bed dryer, see figure 2.6. From the bottom, cooled air is blown in to dry and cool the product. This also causes the powder to float a little, which together with the vibrations make sure that the product moves towards the output side.
The dryer also has thresholds, which slow down the product and ensure that it has a proper drying time. Another aspect that can be controlled, are the separate drying temperatures in each of the four sections of the dryer. Something of note, is the possible additional cooling that needs to occur, depending on the outside temperature (if it is too high, the air can contain more vapour, which needs to be condensed to avoid compromising the product quality). Therefore, meteorological conditions like humidity can have an impact on the energy consumption.

Finally, the product is transported through the Reimeltsystem (collective name for the transportsystems and storage silos of the powder) to the storage silos, where it is ready to be packaged. The actual transport takes place with pressurized air, i.e. pneumatic transportation.

2.3.2. Air line

The air that is used throughout the drying process, of course has to be transported from the outside. First, through a HVAC installation, the air is transported through a series of filters, a dehumidifier and ventilators. The air needs to be dehumidified, because the drying process is faster when the drying air is less humid. Also, variable humidity can negatively impact product quality, so this needs to be controlled for.
Another step that needs to be taken before the air can be utilized at the tower, is pre-heating, which takes place in a so-called recuperator. Some energy savings are achieved by the re-use of the exhaust air from the tower, whose heat is used to evaporate water from the dehumidifier and heat the air in the recuperator.

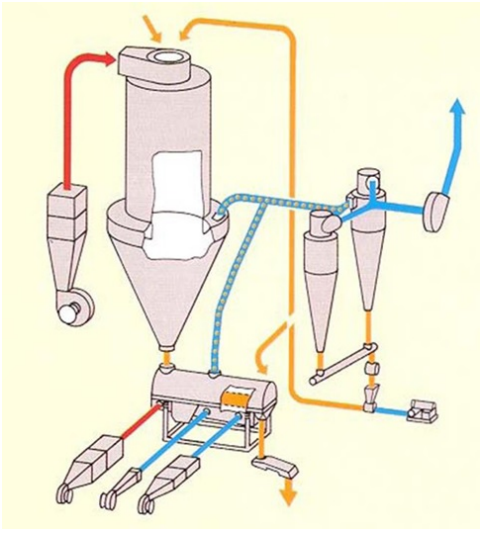


Figure 2.5: Schematic overview of drying tower.

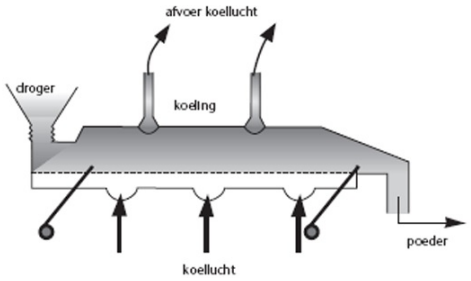


Figure 2.6: Schematic overview of fluidized bed dryer.

After the air is used at the tower, it contains powder particles which need to be retrieved. To this end, two separate installations are used called cyclones ("cyclonen") and bag filters. The former can only retrieve larger particles, but has a larger capacity while the opposite applies for the latter. Therefore, they are used in tandem to adequately remove all particles from the air. The retrieved particles are referred to as *fines return*.
In total, the tower has four cyclones and two bag filters, while the bed dryer has one of each. Furthermore, the air is extracted from the tower/bed dryer through exhaust fans, which transport the air to the cyclones and create low pressure inside the tower. After being cleared from particles, the air is re-used as described before, or simply pumped back outside.

2.3.3. Considerations around drying in tower

To finish this chapter, we would like to zoom in on the drying that occurs after the product leaves the nozzles. Namely, we will summarize the factors that influence the ensuing heat exchange, and therefore the energy consumption of the tower.
As soon as the product leaves the nozzles, the drying process occurs in three stages. During the first stage, the drying speed increases, which leads into the second stage when the drying occurs exponentially fast due to the relatively high amount of water in the product. As all heat is expended to evaporate the water, the temperature of the product does not yet increase (wet bulb temperature). During the final stage, the drying speed decreases as the product contains less water, and therefore start increasing in temperature. It is paramount that the product is cooled as soon as possible, which is why the dryer is located underneath the tower. In summary, here are some important factors that influence the heat exchange.

- **Temperature of drying air**: The bigger the temperature difference, the faster the heat exchange. Though there is an upper limit to prevent heating damage, and a higher exhaust temperature decreases the energy efficiency of the tower.
- **Spraying of product**: The nozzles attribute to the heat exchange by splitting the product in tiny droplets, which increases contact surface between the product and the air. The spraying pattern also makes the droplets easier accessible by all sides from the air. Therefore, these two aspects improve heat exchange.
- **Turbulence of drying air**: The drying is inserted in a swirling motion, which decreases the film coating ("filmlaag", a coating through which heat exchange is difficult) between the droplets and the air.
- **Thermal conductivity of powder particle.**
- **Residence time in tower**: Determined by conical shape of tower and amount of drying air that is blown in.

3

Data Exploration

In this chapter, we will describe the data pre-processing and exploration techniques that we have performed on the data. Data pre-processing is used to transform the raw data from the drying process into a format that is suitable for statistical analysis. Afterwards, data exploration techniques are used to obtain better insight of the data. Notably, we will perform data visualizations that showcase temporal and/or proportional patterns of various parts in the drying process, high-light the problem-areas (such as disproportionally large energy consumption peaks) and we also compare the different profiles of the top ten products in terms of key variables (energy consumption, production volume and time spent in production).
After visualization, we will discuss feature extraction, which is a common data science principle to form new variables as combinations of existing variables [78]. Particularly, we discuss energy efficiency indicators [3] [33] [53] like *SEC*, and use them to compare energy performance between the products. We also perform some time series analysis, to determine the influence of past load values.
Finally, we perform cluster analysis on the energy consumption data, which is a technique often used within energy modelling literature to decompose energy profiles into distinct groups [60] [65]. This may allow one to determine whether there are structural differences between different periods in time.

The data that we will use in this chapter, is displayed in table 3.1. We also note beforehand, that the data we consider for data exploration will eventually serve as test-data for the statistical models we fit later.

Variable	Unit	Description
CIP Step Desc		CIP Step Index Description.
Steam - Dryer CIP	ton	Total Consumed Steam in CIP Procedure.
Energy - Cooling Bed Section	kWh	Total Consumed Electricity in Fluidizer, which is the drying installation underneath tower.
Energy - Spray Dryer 1	kWh	Total Consumed Electricity in Spray Dryer 1 of tower.
Energy - Spray Dryer 2	kWh	Total Consumed Electricity in Spray Dryer 2 of tower.
Gas - Dryer Heater	m^3	Natural Gas Consumption of Heater in Dryer.
Steam - Dryer	ton	Total Consumed Steam in Dryer.
Steam - Dryer Evaporators	ton	Total Consumed Steam in Dryer's Evaporators.
Compressed Air - Dryer	Nm^3	Total Consumed Compressed Air for Powder Drying Transport and Powder Canline
Energy - Drying Elevator	kWh	Total Consumed Electricity in Dryer Elevator, which is used to transport personnel.
Energy - HVAC Niro	kWh	Total Consumed Electricity in HVAC Niro installation.
Energy - Powder Hopper Building	kWh	Total Consumed Electricity in Powder Hopper Building Heating, Silo 11-15.
Product ID		Current Drying Product ID.
Volume - Total Produced	m^3	Total Production of Product.
Energy - Evaporator 1&2	kWh	Total Consumed Electricity of Evaporator 1 and 2.
Energy - UHT 1&2	kWh	Total Consumed Electricity of UHT 1 and 2.

Table 3.1: List of aggregated variables used for data exploration.

3.1. Data pre-processing

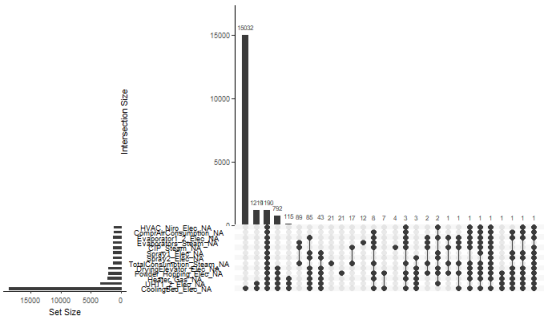
The data was obtained from an integrated data collection platform called *Osi PI*, through a data-link extension in Excel. Aside from expected issues such as missing data and outliers, there is another peculiarity with the data. In order to avoid storing extraneous information, there is a discard of variable values in periods where the variable does not change in any meaningful way. In such periods, the intermittent values are simply interpolated. This means that cumulatively speaking, the system returns equal changes for each time unit.
The following adjustments were made to the data, prior to data exploration.

• Subsequent values were substracted to obtain separate, non-cumulative values for each time unit.

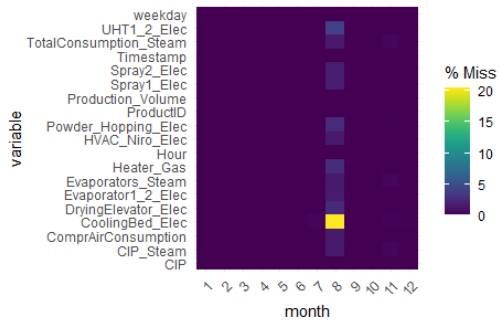
• Logical constraints were imposed on the data. In this case, the raw data could contain negative energy consumption or volume values, and the total steam consumption could be smaller than the sum of its sub-components. Since these occurrences were rare, they were respectively corrected (rather than removed) by setting the negative values to zero, and adding the steam-related subcomponents together.

• Using the provided timestamps, we transformed the data into five different frequencies: minute by minute, hourly, daily, weekly and monthly data.

• The next step involved dealing with the missing data, which we have visualized in figures 3.1a and 3.1b. As can be seen, the missing data rate is generally very low, with the maximum missing amount being less than 20.000 for the cooling bed variable among approximately 600.000 data points (i.e. missing rate of 3.2%). Looking at the common patterns of missing values in figure 3.1a, we see that most often only a single variable is missing at any given time. Another thing of note from figure 3.1b, is that almost all missing values occur during August, which means that the missing data mechanism is likely related to the plant shutdown that routinely occurs during that month.
We used the *R* package *mice* (Multivariate Imputation by Chained Equations) [75] to impute the missing values, which performs the imputation in an iterative manner. In other words, as a Gibbs-sampler it tries to estimate the joint distribution of the set of partially observed variables through separate conditional distributions for each variable. More specifically, let $Y = (Y_1, \cdots, Y_{p_1})$ be the set of partially observed variables with missing values, let Y_j^{obs}, Y_j^{mis} be the respective parts of each such variable containing the observed and missing values, and let $\dot{Y}$ be the currently imputed set of variables. Initially, the missing values Y_j^{mis} are imputed with random draws from Y_j^{obs}. Consequently, in each iteration the missing values Y_j^{mis} are imputed with a separate model $P(Y_j^{mis}|Y_j^{obs}, Y_{-j})$ for this variable, where this model is fitted on those observations with observed values for variable Y_j. For these observations, it also uses all the other variables denoted with Y_{-j}.
The specific imputation model that we have chosen is the CART model as described in [23], where it was shown that CART models can preserve non-linear and interactive effects of the data while requiring minimal input from the user. Furthermore, as a non-parametric method it is capable of dealing with unbalanced factor variables, which is the case for our data. In the above notation, $P(Y_j^{mis}|Y_j^{obs}, Y_{-j})$ becomes a regression tree, and the imputed value for an observation in Y_j^{mis} is randomly chosen from the set of *donors* in the terminal node/leaf that it is assigned to (i.e. each leaf is a subset of Y_j^{obs}). Additional parameters included the amount of imputed datasets m and maximum amount of iterations l, which were chosen as 5 and 10 respectively. The multiple imputations were then averaged to obtain the final imputed dataset. The above described CART implementation in MICE is summarized in Algorithm 1.



(a) Missingness patterns of data.



(b) Missingness patterns over time.

Figure 3.1: Missing data visualization.

Algorithm 1: Implementation of CART in MICE

Data: Partially observed variables $Y = (Y_1, \cdots, Y_{p_1})$, Fully observed variables $X = (X_1, \cdots, X_{p_2})$, amount of iterations l, amount of imputed datasets m
Result: m imputed datasets for Y
for $j \in \{1, \ldots, p_1\}$ **do**
  Let $\dot{Y}_j^0$ be the initial imputations randomly drawn from Y_j^{obs};
Let $\dot{Y}^0$ be the initial fully imputed data matrix;
for $j \in \{1, \ldots, p_1\}$ **do**
  (a) Fit a CART model on $D = [\dot{Y}^0, X]$, limited to observations contained in Y_j^{obs};
  (b) For members in Y_j^{mis}, find the leaf from the fitted model in step (a) that they ended up in;
  (c) For members in Y_j^{mis}, randomly select a value from the leaf in step (b);
  (d) Replace $\dot{Y}_j^0$ with the sampled values in step (c)
Repeat steps (a)-(d) for l iterations, producing sequentially imputed datasets $\dot{Y}^1, \cdots, \dot{Y}^l$. Let $\dot{Y}_1 = \dot{Y}^l$ be the first imputed dataset;
Repeat all previous steps m times;
Return m fully imputed datasets $\dot{Y}_1, \cdots, \dot{Y}_m$

To show convergence of the method, we show traces of the mean and standard deviation of the imputed variables across all iterations in appendix A. Note that we had to perform the imputations separately for smaller subsets of the data, as it was too computationally expensive to perform it for the full data. According to [75], convergence is achieved when the traces intermingle, the variance between traces is approximately equal to the variance within a single trace, and if the traces are free of trend. Upon inspection, most of the imputed values seem to fit these criteria. Finally, we compared the quantiles and mean between the original and imputed values, and found that the differences were insignificant.
• To attain the total energy consumption, we need to convert all the units from the different energy carriers to a common measure. The common measure in this case is the often used energy measure called *Joules* (J). In this case, we convert every different unit to Joules using the conversion factors displayed in table 3.2 (i.e. we multiply the conversion factors with the old units to obtain the consumption in Joules), and then add them up.

Energy Carrier	Conversion Factor	New / Old Unit
Natural Gas	0.03165	GJ/Nm3
Steam	2.609178	GJ/ton
Electricity	0.0036	GJ/kWh
Compressed Air	0.000531	GJ/m3
Chilled Water	0.002275	GJ/kWh(th)
Hot Water	0.318819	GJ/m3

Table 3.2: Conversion factors for the energy carriers.

• To calculate (or we should rather say approximate) the total energy consumption, we add the following variables from table 3.1:

1. Electricity: Everything aside from the cooling bed section, dryer elevator and powder hopper building. The former is excluded because it contains too many missing values, and the latter two are not directly production-related.
2. Gas: Natural gas consumption of heater in dryer.
3. Compressed Air: Consumed compressed air for powder drying transport and powder canline.
4. Steam: Total consumed steam in dryer, the other steam variables are excluded because they are part of it. Moreover, since they were measured from different starting points, they can not be separately utilized.

• We also calculate the total energy separately for the UHTs, evaporators and drying tower:

1. UHTs: Simply its electricity consumption.
2. Evaporators: Its electricity and steam consumption.
3. Tower: Gas consumption of heater and electricity consumption of spray dryers.

• The time range of the data considered for data exploration, is between the 15th of July 2020 and 12th of September 2021. At this stage of the project, this was the maximum period that had (largely) available data for all the variables.

3.2. Feature extraction

Feature extraction refers to the process of forming new variables from the existing variables, whether it is through combinations, transformations or other means. This is an important step in preparing data for model fitting, as new variables defined this way could concisely contain information that would not be possible with the original variables. This can obviously also aid with data-visualization, as visualizing high-dimensional data tends to be difficult. We now explain the additional variables that we have defined.

• As energy load values usually depend on the specific time period [21], we have added time-related variables such as hour, weekday and month. To formally assess the influence of these variables on the energy consumption, we use the Kruskal-Wallis rank sum test [44]. This is a non-parametric test that allows to study the relationship between numeric and categorical variables containing more than 2 levels, see Appendix B.1 for a precise description. We summarize the results in table 3.3, where we can indeed see that generally the time variables have a non-trivial influence.

Frequency	Variable	p-value	Frequency	Variable	p-value
Minute	Hour	***	Hourly	Hour	0.9175
	Weekday	***		Weekday	8.353E-7
	Month	***		Month	***
Daily	Weekday	0.8696	Weekly	Month	0.1378
	Month	2.275E-14			

Table 3.3: The Kruskal-Wallis test applied to test relationship between time-related variables and total energy consumption at a significance level of $\alpha = 0.05$, where the p-values are shown. Cells with *** indicate that the p-value is smaller than 2.2E-16.

• We also considered previous consumption values, where we used the autocorrelation function [15] for a time-series X_t and its shifted counter-part X_{t+h}:

$$\gamma_h = \hat{r}(X_t, X_{t+h}), t = 1, ..., n$$
$$\hat{r}(X_t, Y_t) = \frac{\sum_{t=1}^{n}(X_t - \bar{X})(Y_t - \bar{Y})}{\sqrt{\sum_{t=1}^{n}(X_t - \bar{X})^2 \sum_{t=1}^{n}(Y_t - \bar{Y})^2}}, \tag{3.1}$$

For the lower frequencies, we did not find any significant correlations with the past recent values that occurred. At the hourly frequency, we did strangely find significant peaks at lags of 8019 and 8901, which corresponds to approximately 334 and 371 days. Considering the time distance involved, it was not sensible to use these past values as variables (especially considering that higher lag sample correlations are based on smaller samples). Furthermore, at the weekly frequency we found sensible significant peaks at lags 1,2 and 53, with corresponding correlations of 0.605, 0.351 and 0.311. Again, we had to discard the higher lag value for similar reasons, though if there were sufficient data we could attribute this to a seasonal pattern.

• Another way to take the past values into account, is to calculate the peaks/troughs over the recent values. In our case, we tried to assess the relationship between the peaks/troughs and the total energy consumption through the sample correlation from equation 3.1.
In terms of peak values, we did not detect any significant positive correlations at any frequency while varying the interval size. In terms of trough values, we did find a maximum correlation of 0.173 and 0.464 over an interval of 140 hours and 6 days for the hourly and daily frequency respectively.

• As a widely used energy efficiency indicator [3], the Specific Energy Consumption (SEC) aims to quantify the amount of required energy per unit of some useful output. We will mainly use it in the context of total volume production, i.e.:

$$SEC_{prod} = \frac{\text{Total Energy}}{\text{Total Production}} \tag{3.2}$$

In terms of energy consumed, we will consider total consumption and those of the specific energy carriers. In Section 3.3 we will provide some visualisations, mainly to show the differences in SEC between different product types.

3.3. Data visualization

To begin, we would first like to visualize the energy consumption peaks as we have described before. Figure 3.2 shows the total energy and production levels, along with the separate consumption patterns for the different drying process installations. Of note is that the evaporator and UHT show no peaks, so the peaks seem to mainly correspond with the drying tower.

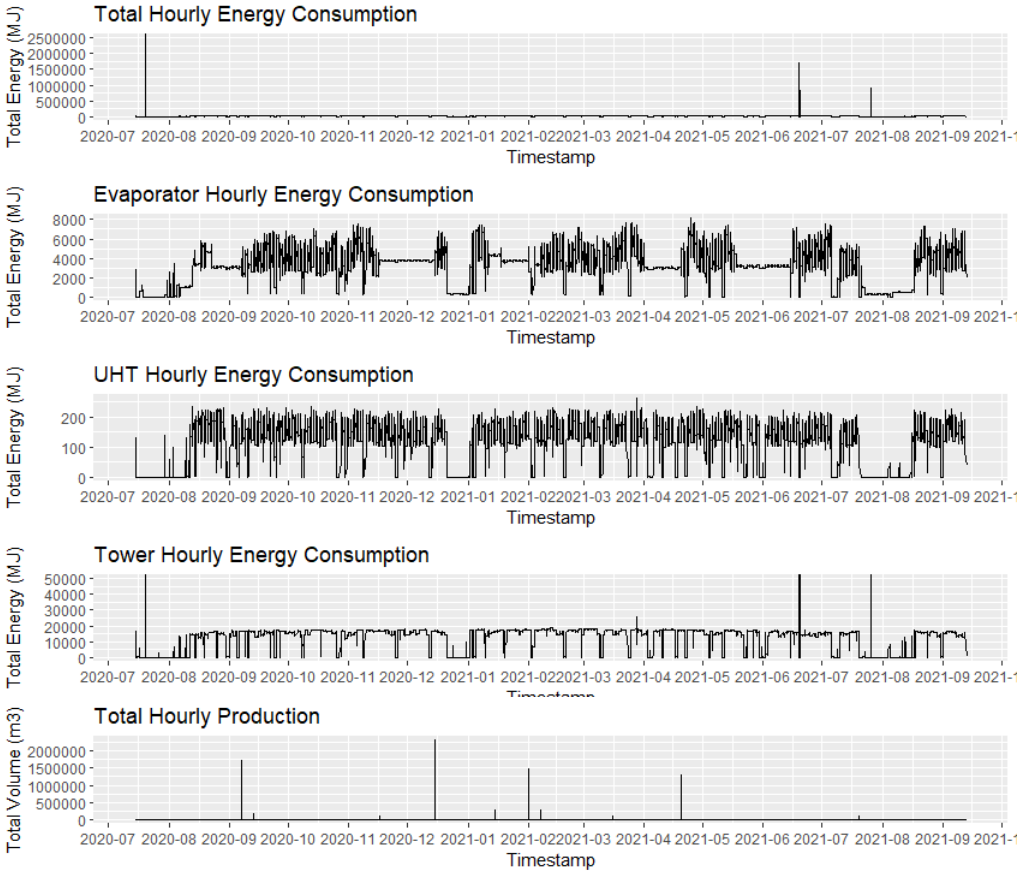


Figure 3.2: Energy peak visualization.

The production graph also displays peaks, but they do not seem to correspond with the energy peaks. As of yet the most probably cause for these peaks are sensor malfunctions, so before fitting models we should remove these outliers, as we will discuss in chapter 4.

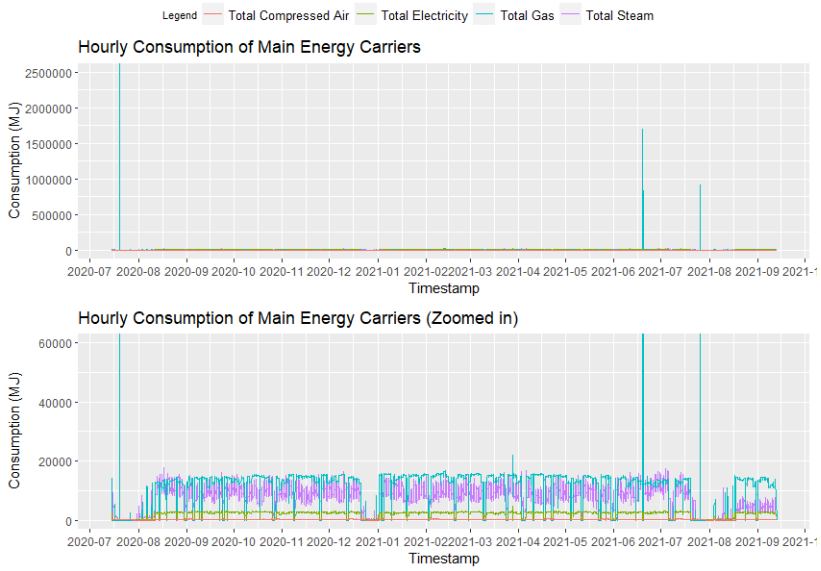


Figure 3.3: Hourly energy consumption of main energy carriers.

To get a better idea of where the peaks seem to originate from, we have displayed a similar plot in figure 3.3 but we

instead show every main energy carrier separately. It is clear that the electricity and gas consumptions account for the peaks. Further things of note are the smaller variance of the gas consumption compared with the steam consumption, despite being of higher level (see figure 3.5). Also, there seems to be an additional shut-down of the plant during the end of the year, which is probably holiday-related. Finally, figure 3.4 shows a close-up view of some differently-sized peaks. Again, this seems to reinforce the point that the peaks between energy-carriers and volume do not have to correspond with each other.

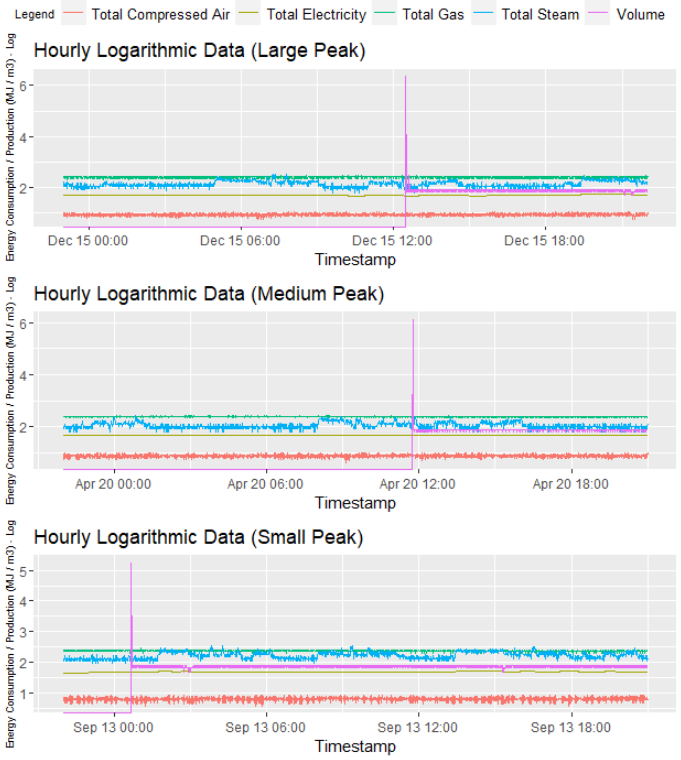


Figure 3.4: Hourly total energy and volume displayed for several peak sizes (log scale).

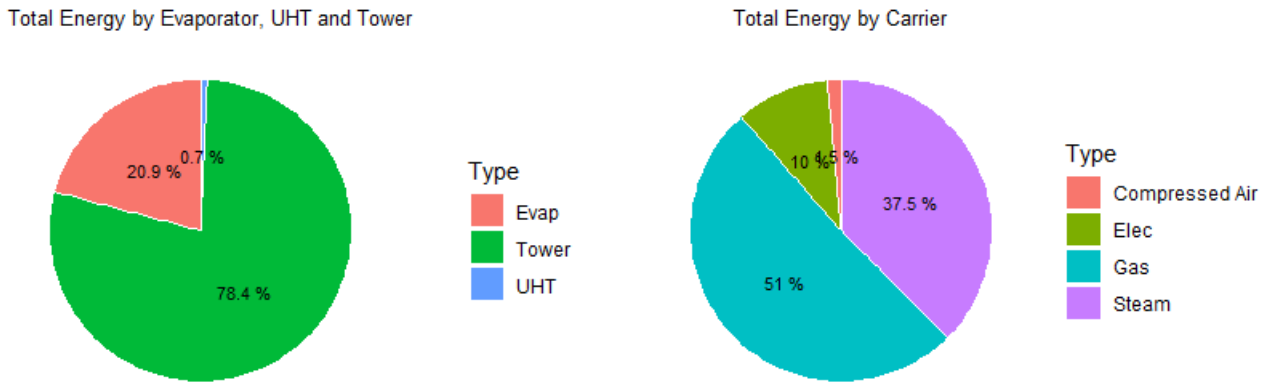


Figure 3.5: Proportions of total energy consumption.

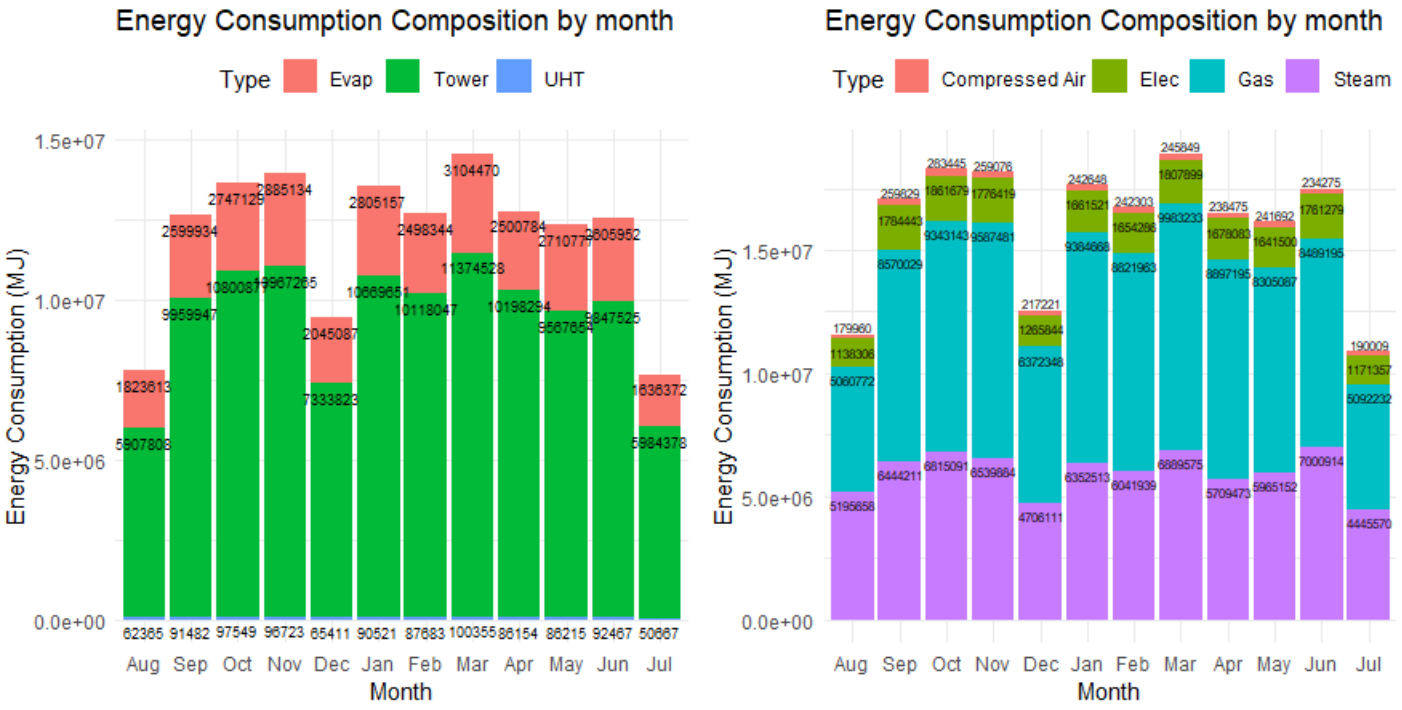


Figure 3.6: Proportions of total energy consumption over time.

Figure 3.5 shows the proportions in total energy by the drying installations and energy carriers. As mentioned before, it is clear that the tower makes up for most of the energy consumption, followed by the evaporators and UHTs. Furthermore, we can see that gas makes up for approximately half of the total energy, followed by steam, electricity and compressed air. Figure 3.6 shows the same proportions over a monthly time-scale.
As Abbott produces many different products, it is also necessary to visualize the most impactful/important product types, so that we may place our focus on them throughout the project. Therefore, in figure 3.7 we have shown the total energy, volume and time in production (i.e. CIP = idle) by product type (the plant shut-down periods were removed before-hand). We note that all ten displayed product types were simultaneously the most impactful for these three variables. For the remainder of this section, we only consider those periods where one of these ten products are in production.

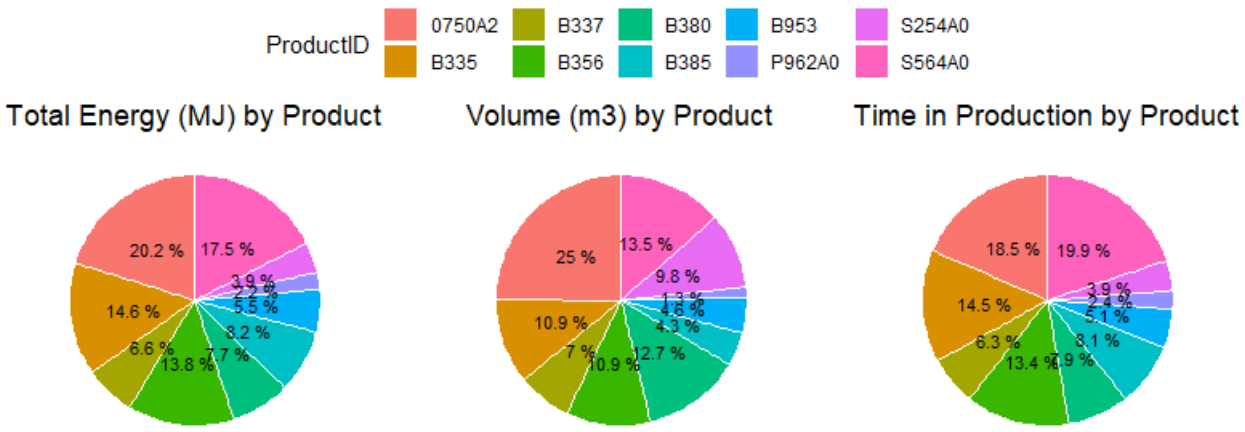


Figure 3.7: Proportions of main characteristics for the top 10 products. Plant shut-down periods are excluded.

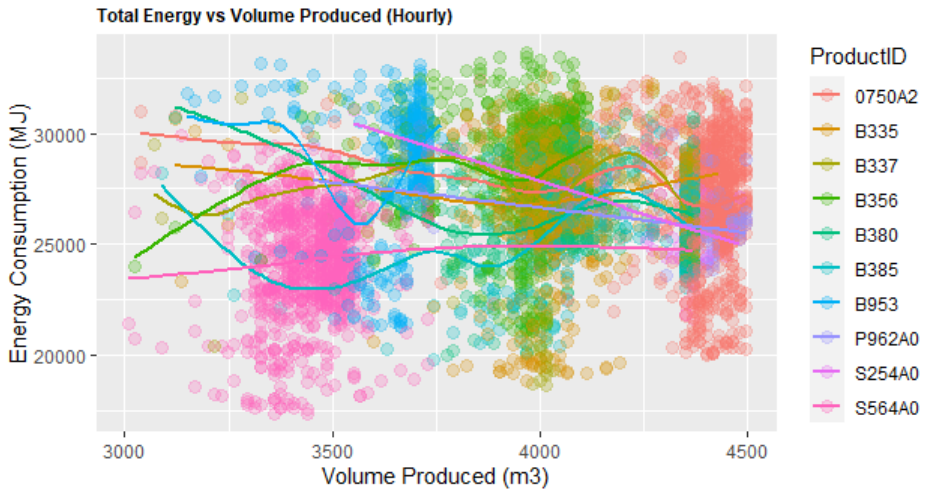


Figure 3.8: Scatterplot between hourly total energy and volume, coloured by the top 10 products. The lines are smoothed splines.

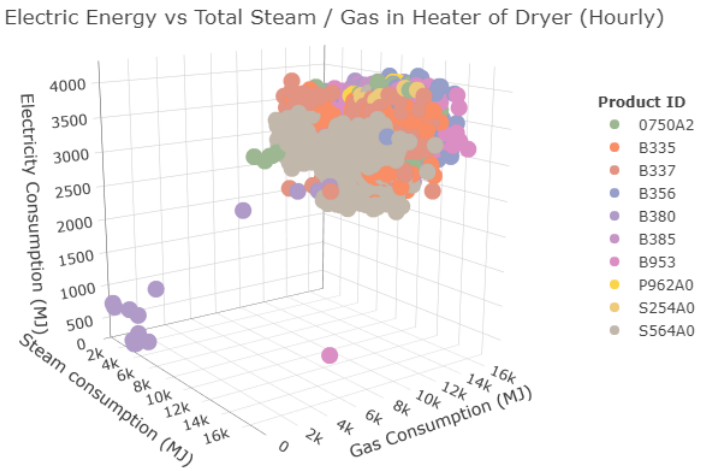


Figure 3.9: 3D-scatterplot between the main energy carriers, coloured by the top 10 products.

Figures 3.8 and 3.9 give an indication of the variance when plotted against several variables. What we mainly wish to show here are that interactions are at play, i.e. the dependencies on the shown variables vary by product. For example, *B953* seems to show a relatively sharper increase in energy consumption when volume is increased. Another thing of note from 3.9, is that the products are separated in quite distinct groups when plotted against the most important energy carriers.

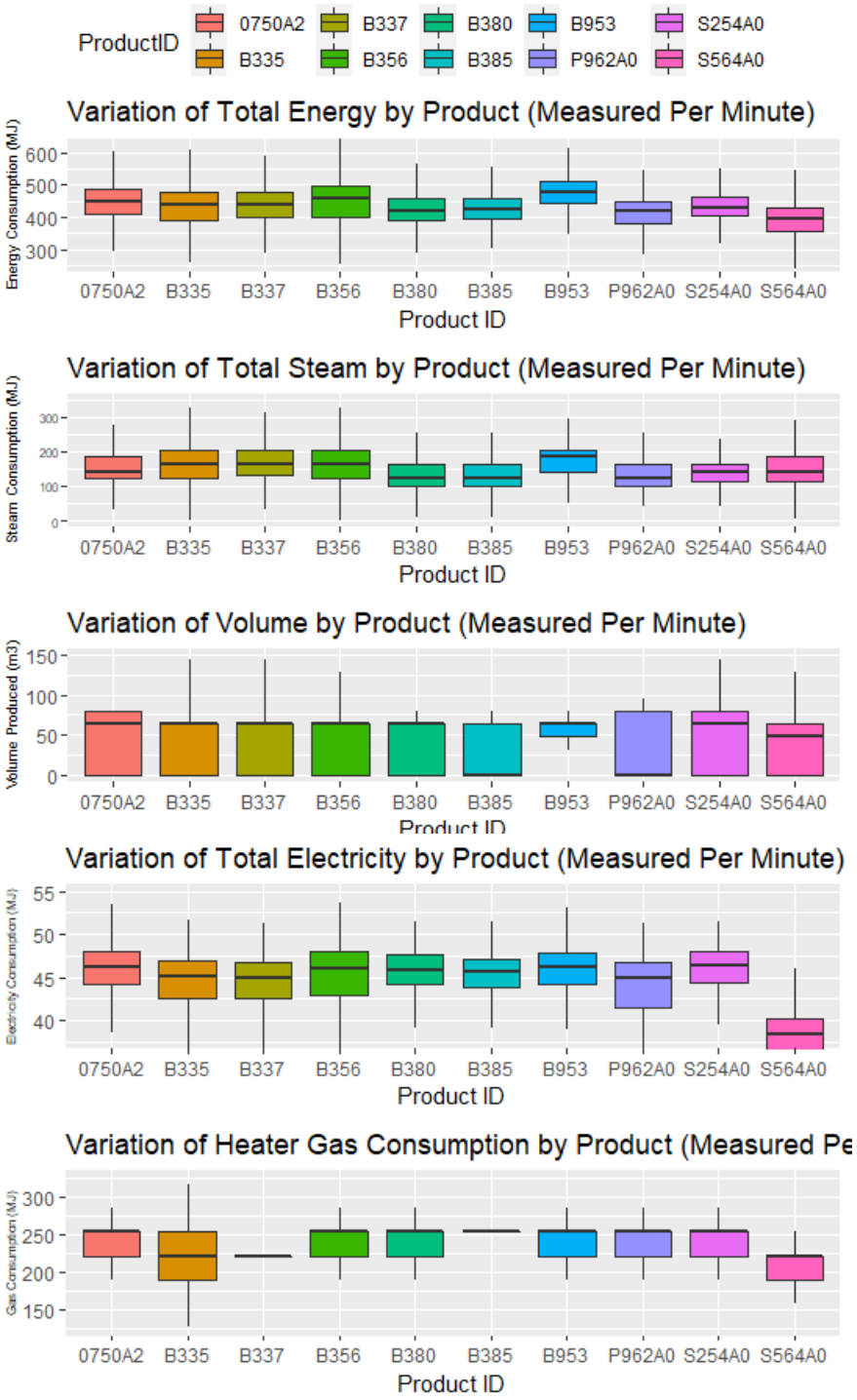


Figure 3.10: Energy consumption and volume variation for the top 10 products.

To obtain a better idea of the variation per product, figure 3.10 shows boxplots of the main energy carriers (aside from compressed air) and volume separately for the top products. In terms of total energy and steam consumption, the consumption patterns across products is quite homogeneous. For electricity and gas consumption however, we can see larger differences. Namely it seems that *S564A0* has the lowest values for these energy carriers.
Volume variation is also mostly the same for nearly all products, with a clear exception being *B953* which has a significantly lower variance.

Finally, we visualize the SEC values separately for each energy carrier and product in figure 3.11. The variation across different energy carriers and products seem to again provide a homogeneous picture, which highlights the *SEC*'s merit

as a comparative energy-performance measure. We also note that in terms of total energy, the highest SEC is clearly obtained by *B953*, while the lowest SEC are obtained by *0750A2*, *S254A0*, *P962A0*.

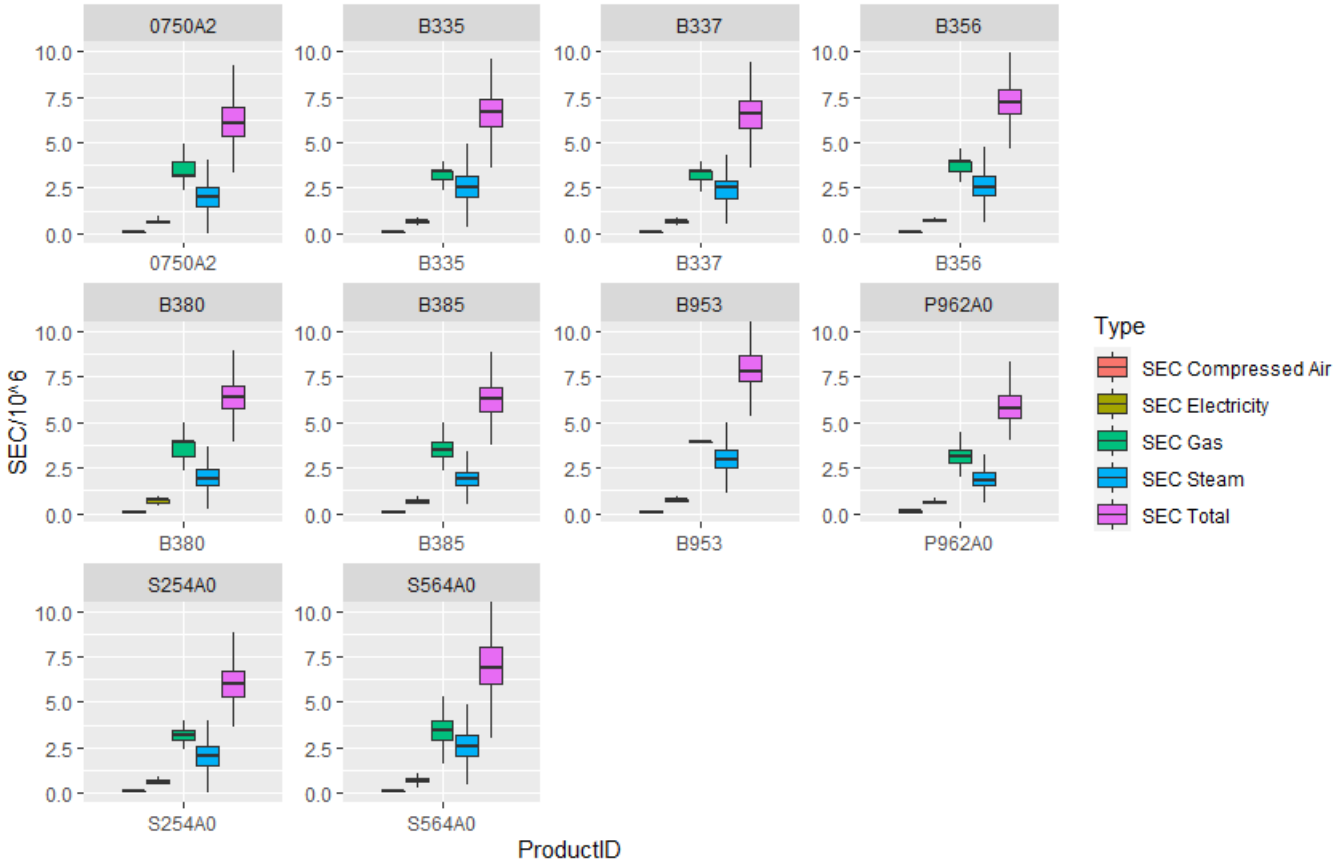


Figure 3.11: SEC visualization for each energy carrier and product separately.

For additional visualizations, see appendix A.2.

3.4. Cluster analysis

Another common data exploration technique, is cluster analysis. In the context of energy load modelling this is commonly applied to split historical data into different groups [60] [65], where the data in each group are characterised by some unique pattern. The reasons for doing this are two-fold: Firstly, better insight into the data can be obtained when each or some of the groups can be interpreted. As an example, we might imagine energy load values to be distinctly different between weekdays and the weekend, which we would expect to be reflected in different groups.
Secondly, clustering the data might be beneficial for model fitting. The idea is that since each group will contain their own specific patterns, a model fitted separately to a group may be better able to capture those specific patterns. Therefore, the general procedure is to classify new data to a group (for example with the use of some nearest neighbour routine), and to then predict the outcome of interest with the group-specific model. Among the predictive models that we will consider in later chapters, we will also study the effect of this procedure.

Several aspects need to be defined, before a cluster analysis can be performed. First of all, we need to define the elements, i.e. the kind of data that needs to be clustered. In our case, we defined as elements the daily total energy profiles, each of which consisted of 24 hourly observations. Therefore the clusters represent the patterns inherent to different days, based on their hourly total energy consumption throughout the day. In essence, we try to cluster time series rather than single points. As described in [63], the considerations underlying both elements are essentially identical, at least when the different time series elements are univariate and perfectly aligned in time (as is the case for our data).
Secondly, a distance function is needed to measure dissimilarity between different elements. Let $X_i = \left(x_{i1}, ..., x_{ij}, ..., x_{i24}\right) (i = 1, ..., N)$ denote the time series for day i, then we simply use the Euclidean distance between different time series elements

X_i and $X_{i'}$:

$$d\left(X_i, X_{i'}\right) = \sqrt{\sum_{j=1}^{24}\left(x_{ij} - x_{i'j}\right)^2} = \|X_i - X_{i'}\|_2 \tag{3.3}$$

Some more advanced distance measures can also be used for time series clustering, as described in [63]. However, these are mainly used when the different time series elements are misaligned and/or have different lengths, neither of which is the case for our data. Once the distance measure is defined, the pairwise distances are summarized in a distance matrix D where $D_{ij} = d(X_i, X_j)$, which is then provided as an input.
The other aspects that need to be defined depend on the clustering algorithm, in this case we consider the partitional *K-means* / *medoids* and agglomerative hierarchical algorithms [36]. The partitional algorithms require the amount of clusters k, specific centroids (cluster centers) and amount of repetitions to be chosen. Meanwhile, hierarchical algorithms do not require any of these specifications, and instead need a measure of inter-group dissimilarity. Since it is not clear before-hand how these aspects should be defined, it is necessary to evaluate the clustering procedure for different combinations of these parameters. For an explanation on how these algorithms work we refer the reader to appendix B.2.

As for the parameters that were varied, we considered k between 5 and 20, five different inter-group dissimilarities ("ward.D", "ward.D2", "complete", "average" and "mcquitty") and three centroid definitions (medoid, mean and median). In terms of fixed parameters, we considered 30 maximum iterations and three repetitions for the partitional algorithms. We also considered the choice of standardizing the data through z-score normalization:

$$\frac{X_i - \hat{\mu}}{\hat{\sigma}}, \tag{3.4}$$

where the estimates for the mean and standard deviation are with respect to the values inside a single time series element. This turned out to be a necessary step, as the original data contained anomalous outlying profiles that could adversely affect the clustering algorithm. Indeed, when trying to cluster without this pre-processing step, many single-element groups consisting of the outliers would end up being formed. For similar reasons, we omitted the "single"-linkage inter-group dissimilarity measure, as it favoured single-element groups.
To evaluate the formed clusters and pick the appropiate model, we have considered two different *cluster validation indices* (CVI) [61] [24]. Let C_i denote the i-th cluster with size $|C_i|$, K the chosen amount of clusters and assume that $X_i \in C_k$, then the CVIs are defined in equations 3.5 - 3.6:

$$\begin{aligned}\text{Silhouette Index} = \frac{\sum_{i=1}^{N} s(i)}{N} = \frac{\sum_{i=1}^{N}\left(\frac{b(i)-a(i)}{\max\{a(i),b(i)\}}\right)}{N}\ (s(i)=0 \text{ if } |C_i|=1), \text{ where}\\ a(i) = \frac{1}{|C_k|-1}\sum_{X_j \in C_k, i\neq j} d(X_i, X_j) \text{ and}\\ b(i) = \min_{k\neq l}\frac{1}{|C_l|}\sum_{X_j\in C_l} d(X_i, X_j)\end{aligned} \tag{3.5}$$

$$\begin{aligned}\text{Dunn Index} = \min_i\left\{\min_j\left(\frac{\delta(C_i,C_j)}{\max_{1\leq k\leq K}\triangle_k}\right)\right\}, \text{ where}\\ \delta(C_k, C_l) = \min_{X_i\in C_k, X_j\in C_l} d(X_i, X_j)\ \text{ (Inter-cluster distance)}\\ \triangle_k = \max_{X,X'\in C_k} d(X, X')\ \text{ (Cluster compactness)}\end{aligned} \tag{3.6}$$

In essence, the CVIs measure the ratio between separation (how well separated are the clusters from each other) and compactness (how close are the elements within the same cluster). To obtain good clusterings, these indices should therefore be maximized. In table 3.4 we summarize for the partitional and hierarchical algorithms the configurations that give the highest scores, separately for each CVI. Note that practically the only difference between K-means and K-medoids is the centroid, which are respectively the mean and medoid (i.e. an original time-series optimally selected as a cluster center, see appendix B.2). Therefore, both of these algorithms are already reflected in the choices for the centroid.

Method	Parameters			CVI	
	k	Centroid	Linkage	Silh	Dunn
Partitional (Silh)	6	Mean	X	0.107	0.203
Partitional (Dunn)	15	Mean	X	0.0986	0.277
Hierarchical	17	X	Average	0.0865	0.351

Table 3.4: Methods optimized according to CVI scores, the lack of parentheses indicates that both CVIs were optimized simultaneously.

To ease data visualization and cluster interpretation, we will choose the configuration that gives the lowest amount of clusters. To that end, we display several characteristics of the clusters in tables 3.5 and 3.6. Namely, we show the relative amounts of maintenance required and the proportions across all the products and time-related variables. Table 3.5 seems to show that there are some differences between the clusters, with the differences being most pronounced for the most energy-intensive products. However, there does not seem to be any relation between the products and amount of maintenance required. In other words, when a cluster contains the most or least maintenance periods, it does not necessarily mean that it will have the most or least of certain products.
Table 3.6 shows similar proportions, but for the time-related variables *weekday* and *month*. The differences are somewhat smaller than expected, as in previous works [65] it was shown that energy profiles usually differed substantially between weekdays, the primary example being between the weekend and the rest of the week. Therefore, from the tabulated proportions we can not conclude that certain clusters contain data from specific periods in time. However, the presence of the shown differences in tables 3.5 and 3.6 in the first place does not discredit the idea that each cluster may still have its own pattern, albeit in a subtle manner.

Cluster	Variables											
	CIP		Product-ID									
	Idle	On	0750A2	B335	B337	B356	B380	B385	B953	P962A0	S254A0	S564A0
1	0.87	0.13	0.17	0.078	0.017	0.098	0.068	0.099	0.02	0.006	0.046	0.315
2	0.92	0.08	0.215	0.11	0.1	0.1	0.019	0.035	0.03	0	0.005	0.259
3	0.85	0.15	0.25	0.096	0.037	0.12	0.031	0.024	0.04	0.042	0.015	0.228
4	0.94	0.06	0.2	0.19	0.029	0.099	0.076	0.04	0.074	0.018	0.032	0.165
5	0.97	0.03	0.19	0.158	0.019	0.145	0.072	0.09	0.015	0.021	0.014	0.225
6	0.93	0.07	0.25	0.052	0.071	0.1	0.086	0.11	0.03	0	0.061	0.162

Table 3.5: Proportions of CIP and Product-ID in time over the clusters.

Cluster	Variables																		
	Weekday							Month											
	Mon	Tue	Wed	Thu	Fri	Sat	Sun	1	2	3	4	5	6	7	8	9	10	11	12
1	0.13	0.16	0.16	0.15	0.13	0.12	0.15	0.07	0.12	0.06	0.1	0.07	0.07	0.06	0.09	0.12	0.04	0.07	0.12
2	0.11	0.17	0.13	0.14	0.13	0.14	0.17	0.06	0.02	0.06	0.08	0.08	0.05	0.17	0.14	0.13	0.08	0.08	0.05
3	0.15	0.12	0.22	0.13	0.09	0.15	0.13	0.06	0.09	0.03	0.09	0.03	0.09	0.15	0.18	0.13	0.06	0.04	0.04
4	0.13	0.09	0.12	0.18	0.13	0.13	0.13	0.04	0.07	0.13	0.09	0.1	0.04	0.06	0.15	0.06	0.09	0.07	0.09
5	0.1	0.12	0.05	0.15	0.23	0.17	0.18	0.08	0.05	0.08	0.03	0.12	0.07	0.12	0.18	0.05	0.1	0.08	0.03
6	0.14	0.19	0.14	0.12	0.18	0.16	0.08	0.12	0.04	0.07	0.03	0.05	0.09	0.11	0.14	0.09	0.07	0.08	0.11

Table 3.6: Proportions of weekday and month in time over the clusters.

We give a final graphical representation of the clusters in figure 3.12. Here we have shown the density function for each cluster in terms of total energy, alongside the normalized daily energy profiles. Due to the normalization, the specific pattern that each cluster possesses is now better visible. Clearly, the fourth and fifth clusters contain the highest peaks, and this can be seen from the normalized profiles as these groups contain profiles with two large peaks throughout the day. Cluster 1 also seems to contain at least two peaks, but these are smaller in scale. When comparing the clusters with peaks, another thing of note is that the peaks seem to occur at different times. Finally, when considering the clusters without peaks, note that they seem to show either a downward (cluster 2) or upward (cluster 3) trend.
In conclusion, this visualization seems to indicate that it could be possible to form different clusters with distinct patterns, at least for daily energy profiles comprised of hourly data.

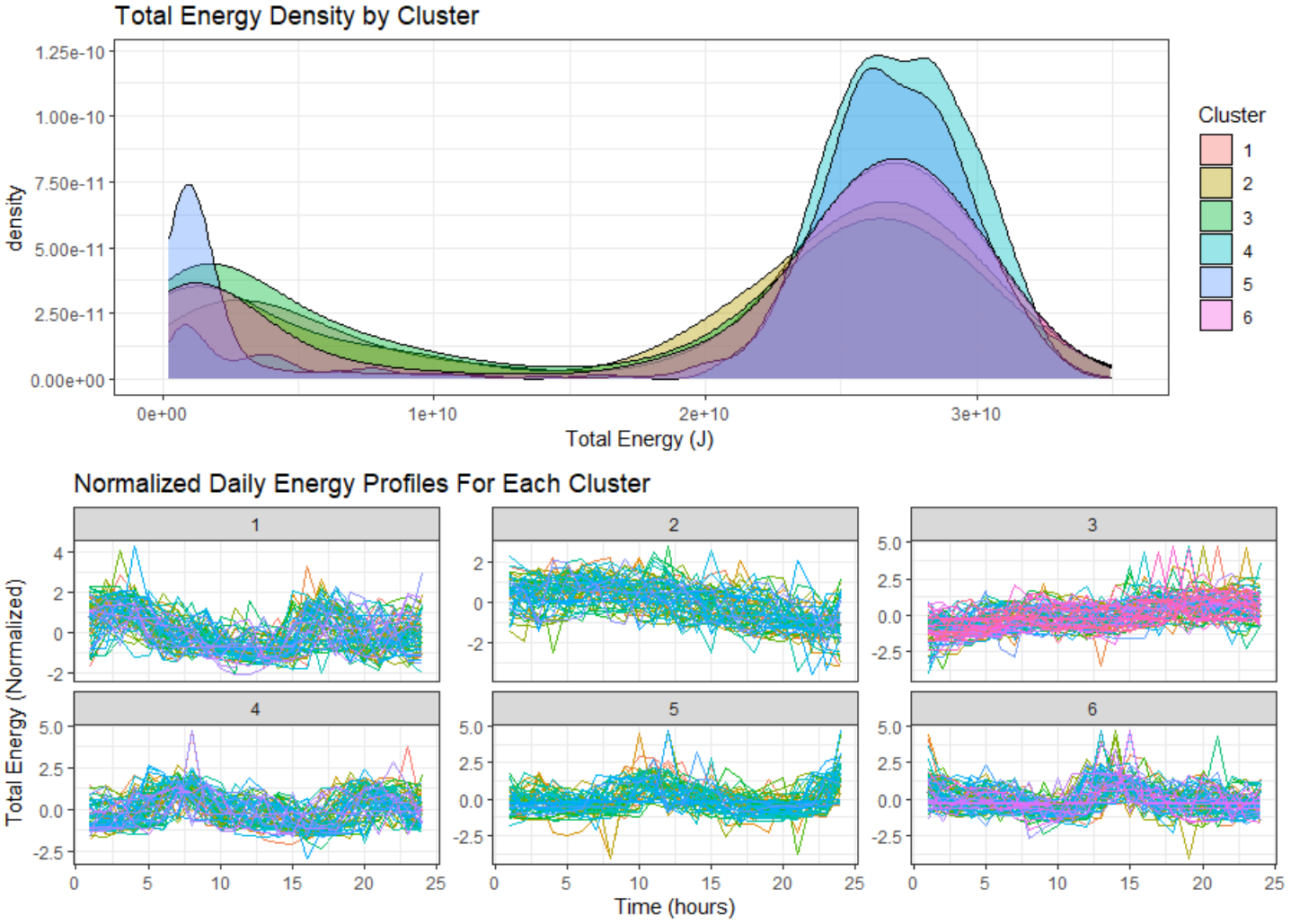


Figure 3.12: Total energy consumption visualized over the different clusters.

4

Basic Model Fitting

As a first step towards obtaining predictive models for the total energy consumption, we mainly consider in this chapter linear regression models. While not generally capable of providing the best performance, they are relatively simple to implement and usually can be interpreted better than more advanced machine learning models.
We will first assess the models primarily from a predictive view-point, i.e. how well can the models predict future values? We assess this from two perspectives: point predictions and prediction intervals.

The structure of this chapter is as follows. First we describe the general procedure for fitting and assessing the models. After that we describe the theory behind linear regression, its regularized variants, the dimension reduction techniques and the prediction intervals. The most accurate linear regression model will serve as a reference model throughout this study. From among all the fitted models, we also incorporate the most accurate ones to calibrate the prediction intervals. Furthermore, we visualize the results in order to gain further insight.
Finally, in order to better portray the results to the stakeholders, we summarize our results and try to address any issues of the used models or methods.

4.1. General Procedure

Before we can start fitting models, we must carefully consider before-hand what the necessary steps should be. The first step that we took, was obtaining a full list of variables which could precisely describe the important parts of the production process. These are production-specific variables such as the outlet temperature of the tower drying air or product density after evaporation. For the full list of variables, along with their descriptions, we refer to Appendix C. During the extraction of the data an issue occurred, namely that the new list of variables were measured at a different frequency than the initial list of variables we looked at in section 3. The latter were measured less frequently over larger time intervals, whereas the former were measured much more often. Therefore, in order to better synchronize the corresponding time-scales, we took 15 minute averages of the new continuous variables, and for the categorical variables we selected their values at the beginning of said periods (which is not a big issue as they generally do not vary quickly across time).

The next step concerns itself with pre-processing the data. Some of the basic steps such as missing data imputation were conducted in the same manner as when we performed data exploration, so we will not go over those again. However, in order to make sure that the data is applicable for model fitting, some additional steps were required. Models like Ordinary Least Squares (OLS) are particularly sensitive towards outlying values and multi-collinearity (i.e. when input-variables are strongly correlated with each other). The results from the model could then become less accurate and more misleading, as we will explain in the next sections. Therefore, we should make sure to study the impact of such aspects when fitting the models.

To detect and remove the outliers, we employed two steps. The first step was a simple univariate approach, where we primarily considered the total energy variable. We calculated its 25-th and 75-th quantiles ($q_{0.25}$ and $q_{0.75}$), and the difference between these two which is called the inter-quantile range: $IQR = q_{0.75} - q_{0.25}$. A value y_j is then considered an outlier, if either $y_j < q_{0.25} - 1.5 \cdot IQR$ or $y_j > q_{0.75} + 1.5 \cdot IQR$.
Outliers may of course also occur in the input variables, and the corresponding observations may then still have a large influence on the model fit (i.e. such observations have high *leverage*). Since there are many input variables, we can not simply use an univariate approach like the IQR-method. Instead, we use *Cook's distance*, which for an observation i is

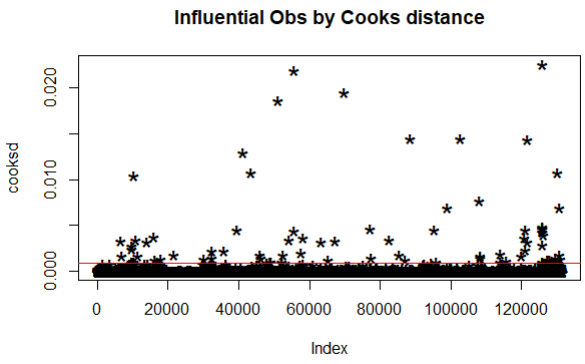
defined as:

$$D_i = \frac{\sum_{j=1}^{n} \left(\hat{Y}_j - \hat{Y}_{j(i)}\right)^2}{p \times MSE}, \tag{4.1}$$

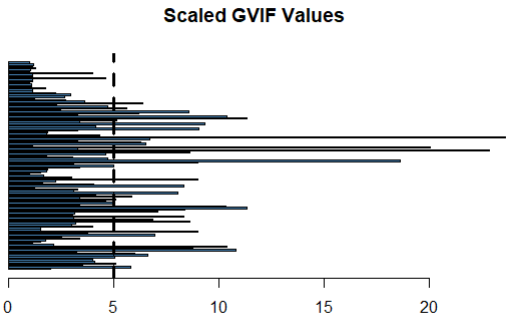
where $\hat{Y}_j$, $\hat{Y}_{j(i)}$ are the j_{th} fitted responses from a linear regression model when all the observations are included in the fit and when the i-th observation is omitted, MSE is the mean-square error and p is the amount of predictors. The Cook's distance simply measures the change in the fitted values $\hat{\boldsymbol{y}}$ when observation i is omitted, so it is a direct measure for its influence. We calculate all the distances, and remove those observations with $D_i >= 100 \cdot \bar{\boldsymbol{D}}$, where $\bar{\boldsymbol{D}}$ is the mean of all the distances. The factor of 100 is certainly not an optimal boundary, we simply chose this value such that visually only the most extreme observations were removed, see figure 4.1a.
Concerning multi-collinearity, we first removed the (nearly) perfect collinear variables, based on the regular linear correlation. Afterwards, we calculated the so-called *generalized variance inflation factors* (GVIF) [30], which are a generalization of the regular VIF for a continuous variable X_j defined as:

$$VIF(X_j) = \frac{1}{1 - R^2_{X_j|X_{-j}}}, \tag{4.2}$$

where $R^2_{X_j|X_{-j}}$ is called the squared R (see equation 4.3c for its definition). The GVIF is meant to also measure multi-collinearity for categorical variables, and it is scaled by the degrees of freedom df as $GVIF_{sc} = GVIF^{\frac{1}{2 \cdot df}}$. The scaling is meant to prevent the value from blowing up for categorical variables (when df is high), and to make its value comparable across all the variables.
To interpret 4.2, it must first be noted that $R^2_{X_j|X_{-j}}$ can be interpreted as the squared correlation between the actual values of X_j and its regressed values from a linear model on all other variables. If the variable X_j is strongly linearly related to other variables, the squared R value will be large, and therefore its VIF will be large. It is therefore proportional to the increase of variance that occurs in the presence of multi-collinearity (see section 4.2), hence its name. A similar interpretation applies for GVIF, see [30].
For VIF values of at least 5 (i.e. $GVIF_{sc}$ at least $\sqrt{5}$), the problem of multi-collinearity is significant [39]. However, when we removed the variables with relatively high values for VIF of 5 and 10, the initial model fits became so poor that the evaluation metrics in equations 4.3 were undefined (due to the exponentiated values becoming too large for R to handle). Since we still wish to assess whether removing multi-collinear variables improves predictive accuracy, we instead removed the most extreme cases with VIF values higher than 25. All the scaled GVIF values are visualized in 4.1b, noting that the scaled GVIF should be seen as the square root of VIF. We can see that many input variables are strongly correlated with each other, even when considering such a high VIF cut-off.



(a) Multi-variate outliers are those with Cook's distances above the red line.



(b) GVIF values for the initial collection of variables, with the GVIF values exceeding the vertical boundary indicating those variables that are strongly correlated with other variables.

Figure 4.1: Additional pre-processing steps.

Furthermore, we had to decide which time-frequency to use. This is difficult to determine at an early stage, because it can not be foreseen how future models or procedures will behave for different time-frequencies. Decreasing the frequency increases the sample size, but may make the computationally intensive algorithms infeasible. On the other hand, increasing the frequency decreases the computational burden, but the small sample sizes will negatively influence the results, especially for those products that have a small sample size to begin with. To pick a time-frequency, we compared the results of the best linear model for frequencies of 15 and 60 minutes, which eventually led us to use the former.
As a final pre-processing step, we need to apply some appropriate transformations. The continuous input variables were

standardized (to make sure that the variables with larger scales do not have higher effects), and we applied a logarithmic transformation to the output variable (to decrease its variance, therefore making it easier to predict). To encode the categorical variables, we considered both ordinal encoding (where each variable is transformed to a vector of integers) and dummy encoding (where a variable is represented by a set of "dummy" variables). We found that the models provide better predictions with the former encoding, and with dummy encoding we also ran into singularity issues when trying to calibrate prediction intervals. These issues are probably caused by the fact that collectively these variables have too many levels relative to the sample size. In other words, dummy encoding produces too many (redundant) variables. Therefore, we proceed with ordinal encoding.

To fit and evaluate the models, we will use a validation-based approach. We split the data into a training set which spawns from the 1st of January 2018 to the 15th of July 2020, and into a test set which continues until the 1st of October 2021. The training set will be used to fit the models, after which we use the fitted models to make predictions of the energy consumption on the test set. Using the actual values we can then evaluate how good the predictions are, where we use separate evaluation metrics for point and interval predictions as we now describe.
In terms of point predictions, we will use the following evaluation metrics for the actual values $\boldsymbol{y}$ and predicted values $\hat{\boldsymbol{y}}$:

$$RMSE(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2} \tag{4.3a}$$

$$MAE(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{y}_i\right| \tag{4.3b}$$

$$R^2(\boldsymbol{y}, \hat{\boldsymbol{y}}) = 1 - \frac{\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2}{\sum_{i=1}^{n}\left(y_i - \frac{\sum_i y_i}{n}\right)^2} \tag{4.3c}$$

Note that we implicitly assume that $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ have been transformed back to the original scale, as we initially applied a logarithmic transformation to the output.
Consequently, this means that the *RMSE* (Root Mean Square Error) and *MAE* (Mean Absolute Error) can be interpreted as the average error on the original scale, with the latter being more resistant against large errors. Furthermore, R^2 can be interpreted as the squared correlation.
After the point accuracy evaluation has been carried out, we should have a good idea of which models perform well in terms of point predictions. For the set of models under consideration, we select the best-performing ones and use it to calibrate prediction-intervals (PIs). We will construct the intervals such that they contain the actual values with an expected probability of 95%. To this end we mainly use computational procedures, which we describe in subsection 4.2.1. For now, we explain how these intervals will be evaluated.
Let $\hat{\boldsymbol{y}}$, $\boldsymbol{y}$ denote the predicted and actual values as before, and let $\{L_i, U_i \mid i = 1, \ldots, n\}$ be the lower and upper endpoints of a PI I_α of level $1-\alpha$, where in our case we have $\alpha = 0.05$ (confidence level of 95%). We will use three measures to evaluate the PIs:

$$\textit{Average Width}(I_\alpha) = \frac{1}{n}\sum_{i=1}^{n}\left(U_i - L_i\right) \tag{4.4a}$$

$$\textit{Coverage}(I_\alpha) = \frac{1}{n}\sum_{i=1}^{n}\mathbb{1}\left(L_i \le y_i \le U_i\right)\cdot 100 \tag{4.4b}$$

$$\textit{Score}(I_\alpha) = \frac{1}{n}\sum_{i=1}^{n}\left(U_i - L_i\right)\cdot\left(\frac{2}{\alpha}(L_i - y_i)\mathbb{1}(y_i < L_i)\right) + \left(\frac{2}{\alpha}(y_i - U_i)\mathbb{1}(y_i > U_i)\right) \tag{4.4c}$$

where $\mathbb{1}(\cdot)$ is the indicator function and metric 4.4c was adapted from [34] (section 6.2). This metric is used because the first two metrics are not enough to adequately assess performance, since they do not account for the specific magnitude of any deviations from the PI. We could for example have two PIs with identical average width and coverage, but the average deviation of the actual values from the outer boundaries could still be drastically different between the two PIs. It is these specific deviations that are measured by 4.4c, conditional on which kind of deviation occurs, and it averages these together with the width. Essentially, the score metric tries to combine the first two metrics. Still, no single metric is exhaustive and every metric should be considered when evaluating PIs. In this case, the optimal PI minimizes the average width and score function, while maximizing the coverage. Also note that as before, the average width and score have to be transformed back to the original scale.

In general, we will predict the overall energy consumption, the energy consumption during active production periods, and the energy consumption during transition/shut-down periods. For the periods of active production, the predictions

will also be assessed separately for the top 10 products in Appendix D, which were found during data-exploration (*S564A0*, *0750A2*, *B335*, *B356*, *B385*, *B380*, *B337*, *B953*, *S254A0*, *P962A0*) and for new products that did not appear in the training data (which also applies to *B953*, but we excluded it from this group since we already evaluate it as part of the top 10).
To select the active production and transitional periods, we used the production step variables for the evaporators, UHTs and tower to find timesteps where these variables simultaneously denote production taking place. Whereas it is straightforward to select the active production periods, finding the transitional periods is slightly more involved. In figure 4.2 we show a graphical overview of how the transitional periods are selected, where we generally distinguish between four kinds of these periods:

1. **Long**: Scheduled shut-down periods, which occur in the span of weeks (which routinely occur yearly during the summer).
2. **Medium**: Regular shut-down periods, which can span from several hours to several days.
3. **Short (same) / Short (diff)**: "Water on tower" periods, where the drying installation is running on water while another production cycle is prepared for the same or different product. These occur over very short periods (mainly because all used energy during these periods is wasted energy).

Note that since the first two periods are shut-down periods, the energy consumption will be much lower during these.

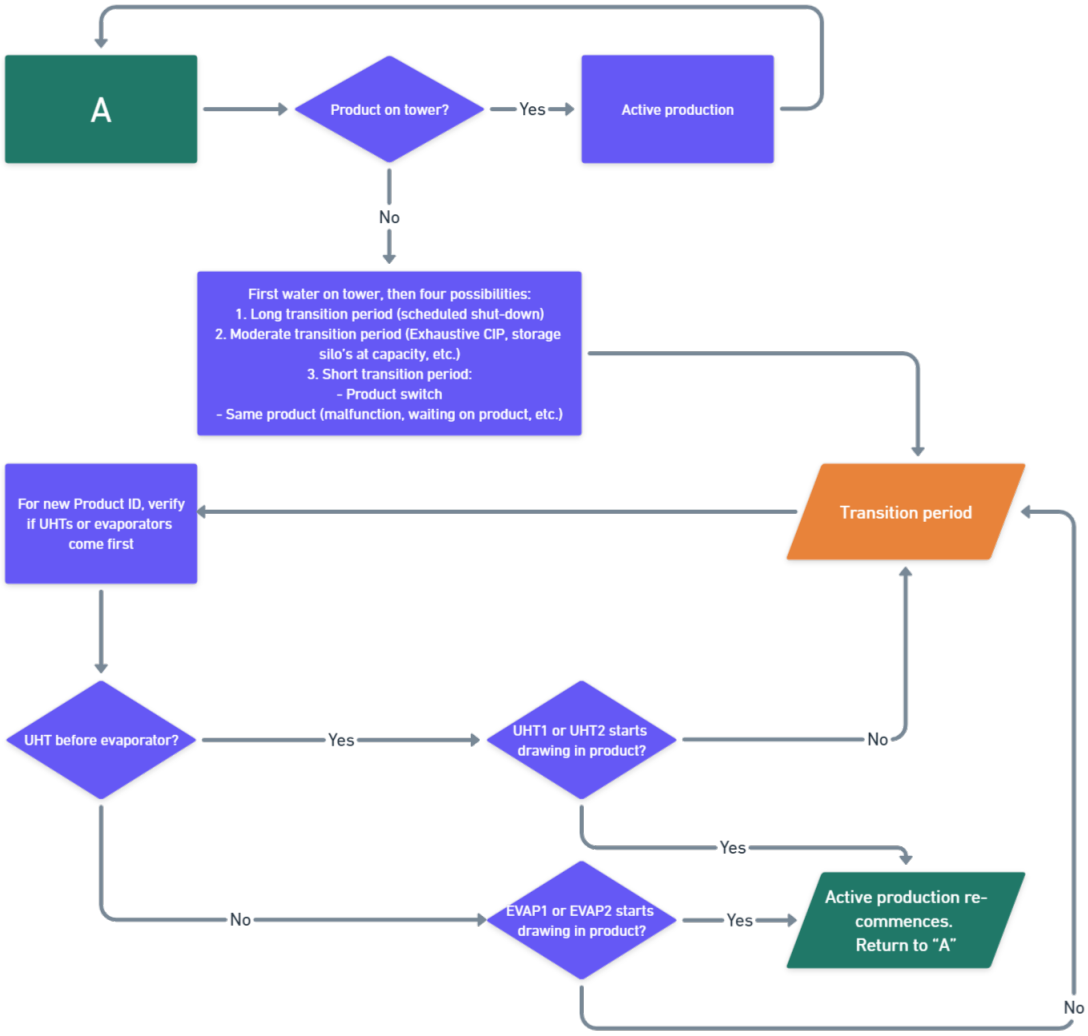


Figure 4.2: A graphical overview of how the transitional periods are selected.

In addition, for the initial OLS model we will first consider several variants of the data (i.e. with/without outliers or multi-collinearity), and whatever configuration returns the best results shall be used for future models. Depending on the period under consideration, i.e. active production, we will also experiment with different variants of the training data as we discuss more precisely in section 4.2.1.2.

Certain models may require so-called *tuning* parameters to be fit, which determine the final shape of the model. For example, regularized regression methods require a value λ, which determines how much it penalizes the size of its parameters. The issue with such parameters is that they can not be analytically optimized, so instead some numeric procedure is required. A popular technique is the cross-validation approach [36]. The idea is that we first split the training data into smaller parts, called *folds*, and define a grid over a suitable range of the tuning parameters. Over each possible combination within the grid, a single fold is repeatedly considered the "test"-data, until the test performance has been assessed on all folds. Hereby, the other folds are used to train the model. All the errors are then averaged to form the cross-validated estimate of the test error $\hat{CV}_{test}$, for the particular grid-point under consideration. The final step is to simply find the grid-point that gives the lowest $\hat{CV}_{test}$, such that the resulting model could be reasonably expected to have optimized its predictive performance for new data.

Finally, we summarize the *R* packages that we have used. For the regularized point predictions, we used *glmnet* [31]. In terms of PIs, we used *quantreg* [42] and *rqPen*[1] for regular and regularized quantile regression, and *conformalInference* [2] for conformal inference.

4.2. Linear Regression

In multi-variate regression problems, we try to approximate some fixed functional relationship between a quantitative outcome variable y and some set of predictor variables $x_1, x_2, ..., x_p$:

$$y = f(x_1, ..., x_p) + \epsilon, \tag{4.5}$$

where p is the amount of input variables, $f(\cdot)$ is a function that can represent any kind of relationship and $\epsilon \sim \mathcal{N}(0, \sigma^2)$ sums up all the error sources. Of course, $f(\cdot)$ is never explicitly known in practice, so some assumptions are required to make the approximation problem more tangible.
Linear regression makes the problem easier, by assuming that $f(\cdot)$ takes some linear, additive form:

$$f(x_1, ..., x_p) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p, \tag{4.6}$$

where $\boldsymbol{\beta} = [\beta_0, ..., \beta_p]$ is the set of parameters which characterizes the problem. The challenge is to find an optimal approximation $\hat{\boldsymbol{\beta}}$ such that some performance criterion is minimized, usually taken to be some distance measure between the actual values y and predicted values $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \cdots + \hat{\beta}_p x_p$. In essence, by constraining $f(\cdot)$ to be of a particular form, the underlying parameter space is also restricted. This makes finding the optimal parameter vector much easier, with popular methods like OLS, Ridge and LASSO even allowing analytical forms for the optimal parameter values, as will be shown in the next sections.
Aside from assuming that $f(\cdot)$ should take some fixed form, linear regression models make some other restrictive assumptions [36] which we summarize below:

1. There must be little or no multi-collinearity among the predictor variables. If this is not the case, the responsible variables should be centered or removed.
2. The residuals should be normally distributed.
3. The residuals must not be auto-correlated.
4. The residuals must show homoscedasticity, i.e. the variance of the residuals should remain approximately constant across time.
5. The predictor variables are non-random.

When the above assumptions are not satisfied, it usually means that the data contains non-linear patterns which linear regression can not account for. As a consequence, hypothetical tests used to test significance of the coefficients are less valid, since the underlying test-statistics use estimates which are based on the above assumptions (namely the variance, see [36]). Despite the strict assumptions and general in-flexibility of linear regression, it still serves as an excellent stepping stone when the goal is to eventually study more complicated models. Not only because a lot of models share many aspects with linear regression, but as linear regression is easy to fit and interpret, it can quickly provide some vital insights of the data. For example, violation of the above assumptions can be visualized to point out any deficiencies of the model, and this can drive the search for more advanced models.

[1] https://github.com/bssherwood/rqpen
[2] https://github.com/ryantibs/conformal

4.2.1. Ordinary Least Squares

Theory

Denote the training data set as $\{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$, where $x_i = (x_{i1}, x_{i2}, ..., x_{ip})^T$ and y_i belong to the ith observation and respectively denote the p predictor variables and outcome variable. OLS seeks to find a parameter combination $\beta = (\beta_0, \beta_1, ..., \beta_p)^T$ such that the residual sum of squares (RSS) is minimized:

$$\begin{aligned} \text{RSS}(\beta) &= \sum_{i=1}^{N} \left(y_i - f(x_i)\right)^2 \\ &= \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j\right)^2 \end{aligned} \tag{4.7}$$

To optimize for β, some matrix-vector notation is required. Let $\boldsymbol{X}$ denote the $N \times (p+1)$ design matrix, containing in every row an input vector (with the first column solely containing ones to represent the intercept), and similarly let $\boldsymbol{y}$ be the N-sized output vector. Then 4.7 can be written as:

$$\text{RSS}(\beta) = (\boldsymbol{y} - \boldsymbol{X}\beta)^T(\boldsymbol{y} - \boldsymbol{X}\beta). \tag{4.8}$$

As this is quadratic in β, finding the optimal value is simply a matter of equating the first derivative to zero:

$$\begin{aligned} \frac{\partial \text{RSS}}{\partial \beta} &= -2\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\beta) = 0 \\ &\Rightarrow \boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\beta) = 0 \\ &\Rightarrow \hat{\beta} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y}. \end{aligned} \tag{4.9}$$

A nice property of the OLS estimator is unbiasedness:

$$\mathbb{E}(\hat{\beta}) = \mathbb{E}\left[\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y}\right] = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\mathbb{E}(\boldsymbol{y}) = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{X}\beta = \beta. \tag{4.10}$$

Note that $\hat{\beta}$ is only an unique solution when $\boldsymbol{X}^T\boldsymbol{X}$ is positive definite and therefore invertible, i.e. when $\boldsymbol{X}$ has full column rank. This is also why multi-collinearity is a serious problem, as it makes $\boldsymbol{X}$ more singular.
The predicted values $\hat{\boldsymbol{y}}$ are then obtained as:

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\beta} = \boldsymbol{X}\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{4.11}$$

Prediction Intervals

Up until now, we have mainly described how linear regression models can produce singular prediction points. However, single prediction points are not sufficient to gauge the predictive performance of a model, as it does not indicate how much uncertainty is involved with these predictions. Rather, one should construct prediction *intervals*, which are designed to contain the true values with a certain chance. Ideally, we would like to construct prediction intervals that not only contain the actual values with a probability of 0.95, but can adjust itself to the input variables in order to produce locally varying intervals. This will allow us to assess in which scenarios the uncertainty is smaller or larger, for example which product has the largest or smallest underlying uncertainty.
We discuss several methods of varying complexity, which are either model-specific (i.e. "analytic" approach is only possible for OLS) or can be generally applied to any regression model.

A) **Analytic**: For this approach we need to assume that we are fitting a OLS model (and therefore that the data adheres to its strict assumptions). If we observe some new feature vector x_{new}, we can obtain a prediction interval for some newly predicted value $\hat{y}_{new}$ as follows. First, denote the (constant) variance of the training data's output variable $\boldsymbol{y}$ as σ^2, the variance-covariance matrix of the least squares estimate $\hat{\beta}$ can then be easily obtained from 4.9 as:

$$\begin{aligned} \text{Var}(\hat{\beta}) &= \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\sigma^2\boldsymbol{X}\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1} \\ \Rightarrow \text{Var}(\hat{\beta}) &= \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\sigma^2. \end{aligned} \tag{4.12}$$

Prediction intervals need the standard error for the sum of new predictions and an additional term ϵ_{extra} representing the additional variance of an observation about its mean, i.e. $\text{SE}(\epsilon_{extra}) = \sigma$. Since $\hat{y}_{new} = x_{new}^T\hat{\beta}$, we can calculate this standard error using 4.11 and 4.12 as:

$$\begin{aligned} \text{SE}\left[\hat{y}_{new} + \epsilon_{extra}\right] &= \sqrt{\sigma^2 + \sigma^2 x_{new}^T\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}x_{new}} \\ &= \sigma\sqrt{1 + x_{new}^T\left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}x_{new}}. \end{aligned} \tag{4.13}$$

The prediction intervals are then defined as:

$$\hat{y}_{new} \pm t_{\alpha/2,N-p-1} \cdot \hat{\sigma} \sqrt{1 + x_{new}^T \left(\boldsymbol{X}^T \boldsymbol{X}\right)^{-1} x_{new}}, \tag{4.14}$$

where $t_{\alpha/2,N-p-1}$ denotes the $\frac{\alpha}{2}$-th quantile for a t-distribution with N - p - 1 degrees of freedom and

$$\hat{\sigma} = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} \left(\epsilon_i - \bar{\epsilon}\right)^2} \tag{4.15}$$

is an unbiased estimate for σ, where N is the training data size and $\epsilon_i = y_i - \hat{y}_i$ are the training residuals.
Note that the interval is only valid when the OLS model assumptions are not violated, so a big disadvantage of this method is that it is sensitive to these violations (which is common for real-life data, in particular for our data as we show in section 4.2.1.2). Furthermore, this method essentially amounts to adding and subtracting a constant to the predictions, so it can only return a non-flexible, fixed-width prediction band.

B) **Bootstrap**: The bootstrap is a computational procedure, that can be used to empirically assess the uncertainty of any statistic of interest. In this case we can use it to numerically estimate the prediction uncertainty, rather than use a theoretical formula like in 4.14. We will adopt the method from [46], which we have slightly modified as we will describe now.
First, it is assumed that the true model $y : \mathbb{R}^p \to \mathbb{R}$ is of the same form as 4.5:

$$y(\boldsymbol{x}) = f(\boldsymbol{x}) + \epsilon(\boldsymbol{x}), \tag{4.16}$$

with $f(\boldsymbol{x})$ denoting the true underlying model and $\epsilon(\boldsymbol{x})$ the noise function. For $f(\boldsymbol{x})$ and $\epsilon(\boldsymbol{x})$ some specific assumptions are made, namely that $f(\boldsymbol{x})$ must be deterministic (it always provides the same output for identical values of $\boldsymbol{x}$) and that it is sufficiently smooth (i.e. continuously differentiable), and that $\epsilon(\boldsymbol{x})$ are i.i.d. for all $x \in \mathbb{R}^p$.
The above model is estimated with a regression function $\hat{\boldsymbol{y}}_N : \mathbb{R}^p \to \mathbb{R}$, which is trained on a training data-set of size N. Likewise, it has some specific assumptions. It must be deterministic and continuous, while converging pointwise to a function $\hat{\boldsymbol{y}} : \mathbb{R}^p \to \mathbb{R}$ as $N \to \infty$ and it must asymptotically have zero bias: $\mathbb{E}\left[\hat{\boldsymbol{y}}_N(\boldsymbol{x}) - f(\boldsymbol{x})\right]^2 \to 0$ as $N \to \infty$ for every $\boldsymbol{x} \in \mathbb{R}^p$. The latter assumption in particular is troublesome, as it assumes that $\hat{\boldsymbol{y}}_N$ will obtain zero training error with sufficient data. However, as we will show when discussing the regularized methods, certain models will always have bias, so this assumption would not hold anymore. Therefore, we relax this assumption and instead simply assume that the bias $\eta(\boldsymbol{x})$ should exist asymptotically for every $\boldsymbol{x} \in \mathbb{R}^p$, i.e.:

$$\eta(\boldsymbol{x}) = \lim_{N \to \infty} \mathbb{E}\left[\left(\hat{\boldsymbol{y}}_N(\boldsymbol{x}) - f(\boldsymbol{x})\right)^2\right] \tag{4.17}$$

must exist.

To form the prediction intervals, all error sources need to be identified and estimated. This can be done by re-writing the output variable $y(x_{N+1})$ for a new observation $x_{N+1} \in \mathbb{R}^p$ as:

$$y(x_{N+1}) = f(x_{N+1}) + \epsilon(x_{N+1}) = \hat{y}_N(x_{N+1}) + \eta(x_{N+1}) + \eta_N(x_{N+1}) + \epsilon(x_{N+1}), \tag{4.18}$$

where $\eta_N(\boldsymbol{x}) = f(\boldsymbol{x}) - \hat{y}_N(\boldsymbol{x}) - \eta(\boldsymbol{x})$ is the model variance noise. From 4.18 it is clear that the total error surrounding the prediction $\hat{y}_N(x_{N+1})$ consists of three parts: $\eta(x_{N+1})$ (model bias), $\eta_N(x_{N+1})$ (model variance noise) and $\epsilon(x_{N+1})$ (sample noise). Therefore, these sources of error should be estimated in order to produce the prediction intervals.
To estimate the model variance noise, we first re-sample the training data with replacement B times, where $B = \sqrt{N}$ as recommended in [46]. On each of the bootstrapped samples, we fit a new model $\hat{\boldsymbol{y}}_{b,N}(\boldsymbol{x})(b = 1, ..., B)$, which in turn provides B bootstrapped predictions $\hat{y}_{b,N}(x_{N+1})$. We then calculate their mean:

$$\hat{\mu}_N(x_{N+1}) = \frac{1}{B} \sum_{b=1}^{B} \hat{y}_{b,N}(x_{N+1}), \tag{4.19}$$

and center the bootstrapped predictions: $m_b = \hat{\mu}_N(x_{N+1}) - \hat{y}_{b,N}(x_{N+1})$. From 4.17 it then follows:

$$\mathbb{E}\left[m_b\right] = \mathbb{E}\left[\hat{\mu}_N(x_{N+1})\right] - \mathbb{E}\left[\hat{y}_{b,N}(x_{N+1})\right] \to_{B \to \infty} \mathbb{E}\left[f(x_{N+1}) - \eta(x_{N+1})\right] - \mathbb{E}\left[\hat{y}_N(x_{N+1})\right] = \mathbb{E}\left[\eta_N(x_{N+1})\right],$$

so m_b can be used to approximate the model variance noise.
The model bias $\eta(x_{N+1})$ and sample noise $\epsilon(x_{N+1})$ can both be estimated at once from the residuals. From the bootstrapped models $\hat{y}_{b,N}$, we can select observations that were not included in the bootstrapped samples, and calculate the validation errors:

$$\text{val\_error}_{b,i} = y(x_i) - \hat{y}_{b,N}(x_i), \tag{4.20}$$

for $1 \leq b \leq B$ and $1 \leq i \leq N$ such that x_i is not in the b-th bootstrap sample. Taking expectations w.r.t. b then gives:

$$\mathbb{E}_b\left[\text{val\_error}_{b,i}\right] \approx \eta(x_i) + \eta_N(x_i) + \epsilon(x_i) \rightarrow_{N\rightarrow\infty} \eta(x_i) + \epsilon(x_i), \tag{4.21}$$

so the validation residuals can be used to estimate the sum of the bias and sample noise. Note that the same applies when the training residuals are used, i.e. $\text{train\_error}_i = y(x_i) - \hat{y}_N(x_i), i = 1, ..., N$ for the full model $\hat{y}_N$. The authors in [46] only used the training residuals.
Therefore, we can use either the validation or training residuals. However, neither is optimal. The training residuals will always underestimate the true error when the model is over-fitting, and the validation residuals will overestimate it since the bootstrapped models use on average two thirds of the training data for training.
This issue has been addressed in [36] (section 7.11), where the authors used the ".632"-estimator to combine both kinds of residuals, with the weights being determined by the degree of over-fitting. Let train_error and val_error be the percentiles of the respective residuals, then we first need the so-called *no-information error rate*:

$$\hat{\gamma} = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left(y(x_i) - \hat{y}_N(x_j)\right)^2, \tag{4.22}$$

which can be interpreted as the average loss when the inputs and outputs are independent. Secondly, we calculate the *relative overfitting rate*:

$$\hat{R} = \frac{\text{val\_error} - \text{train\_error}}{\hat{\gamma} - \text{train\_error}}, \tag{4.23}$$

which goes from 0 to 1 as the degree of over-fitting increases. The aforementioned weights are determined by $\hat{w} = \frac{.632}{1-.368\hat{R}}$, so we estimate $\epsilon(x_{N+1}) + \eta(x_{N+1})$ by:

$$o_{perc} = (1 - \hat{w}) \times \text{train\_error} + \hat{w} \times \text{val\_error}, perc = 1 \ldots 99. \tag{4.24}$$

The subscript $(\cdot)_{perc}$ thus denotes all percentiles between the first and 99th. When there is severe over-fitting, the influence of the training residuals are mitigated.
As a final step, the convolved set between m_b and o_i is computed:

$$C := \left\{m_b + o_{perc} \mid b \leq B, 1 \leq perc \leq 99\right\}, \tag{4.25}$$

which approximates the distribution of $\eta(x_{N+1}) + \eta_N(x_{N+1}) + \epsilon(x_{N+1})$. Its $\frac{\alpha}{2}$-th and $1 - \frac{\alpha}{2}$-th quantiles are then computed, which form the boundaries for the prediction $\hat{y}_N(x_{N+1})$.

The above procedure is summarized in Algorithm 2 (note that for computational reasons, $\hat{\gamma}$ is estimated by sampling rather than through all combinations). Its main advantage is its generalizability, as it can be effortlessly adapted to any regression model. However, due to the many model fits required, it is very computationally intensive. An approach like this would not be feasible for complicated models requiring many computation steps.

C) **Quantile regression**: Quantile regression is closely related to OLS regression, but rather than providing an estimate of the conditional mean, quantile regression estimates the conditional quantiles through separate regression functions $\hat{y}_\tau$, where τ is the quantile-level of interest. A 95% PI can then simply be formed by estimating $\hat{y}_{0.025}$, $\hat{y}_{0.975}$ and $\hat{y}_{0.5}$, which would respectively form the outer boundaries and prediction points. More specifically, the square loss function $L(y, \hat{y}) = (y - \hat{y})^2$ as in equation 4.7 is now replaced by the *quantile* loss function:

$$L_\tau(y) = y\left(\tau - \mathbb{1}(y < 0)\right). \tag{4.26}$$

Likewise, the regression function is assumed to have a linear form, and the goal is to find an estimate $\hat{\beta}_\tau$ such that:

$$\hat{\beta}_\tau = \underset{\beta\in\mathbb{R}^{p+1}}{\text{argmin}} \sum_{i=1}^{N}\left(L_\tau(y_i - x_i\beta)\right), \tag{4.27}$$

such that the i-th τ-level quantile for some input x_i can be estimated by

$$\hat{y}_{i,\tau} = x_i\hat{\beta}_\tau. \tag{4.28}$$

The proof that 4.28 estimates the τ-th quantile is given in appendix B.5.
The main advantage of this method is its flexibility, as it does not have strict distributional assumptions like OLS, and it can immediately provide the PI as output. Furthermore, it it fitted simply by using a different loss function, making it relatively easier to adapt for other predictive models.

Algorithm 2: A bootstrap procedure for estimating PIs

Data: $\boldsymbol{x}_{\text{train}}$, $\boldsymbol{y}_{\text{train}}$, $\boldsymbol{x}_{\text{test}}$, $\alpha \in (0,1)$, regression algorithm $\hat{y}_N(\cdot)$
Result: $(1-\alpha)$% prediction interval $I_\alpha(\boldsymbol{x}_{\text{test}})$ for $\boldsymbol{y}_{\text{test}}$
$N \leftarrow$ number of observations in $\boldsymbol{x}_{\text{train}}$;
$B \leftarrow \sqrt{N}$;
Build B bootstrap samples from $\boldsymbol{x}_{\text{train}}$ with replacement;
Initialize bootstrap sample set $D = \emptyset$;
Initialize validation error set $E_{\text{valid}} = \emptyset$;
for *each bootstrap sample* $B_b, b = 1 \dots B$ **do**
  Build regression models $\hat{y}_{b,N}(B_b)$;
  Obtain centered samples m_b;
  $D \rightarrow D \cup \{m_b\}$;
  For left-over training samples $(x_i, y_i), i = 1 \dots n$ compute $E_i = y_i - \hat{y}_{b,N}(x_i)$;
  $E_{\text{valid}} = E_{\text{valid}} \cup \left\{\bigcup_{i=1}^{n} E_i\right\}$;
Build full regression model $\hat{y}_N(\boldsymbol{x}_{\text{train}})$;
Initialize training error set $E_{\text{train}} = \emptyset$;
for *each training sample* (x_i, y_i) **do**
  Compute $E_i = y_i - \hat{y}_N(x_i)$;
  $E_{\text{train}} = E_{\text{train}} \cup E_i$;
$E_{\text{train}} \leftarrow \left\{\bigcup_{i=1}^{99} E_{\text{train}}\,{}_{\frac{i}{100}}\right\}$;
$E_{\text{valid}} \leftarrow \left\{\bigcup_{i=1}^{99} E_{\text{valid}}\,{}_{\frac{i}{100}}\right\}$;
Compute $\hat{\gamma}, \hat{R}$ and $\hat{w}$;
Compute $o_{perc} = (1-\hat{w}) \times E_{\text{train}} + \hat{w} \times E_{\text{valid}}$;
$C := \left\{m_b + o_{perc} \mid b \le B, 1 \le perc \le 99\right\}$;
Compute $C_{\frac{\alpha}{2}}, C_{1-\frac{\alpha}{2}}$;
Set $I_\alpha(\boldsymbol{x}_{\text{test}}) = \hat{y}_N(\boldsymbol{x}_{\text{test}}) + \left\{C_{\frac{\alpha}{2}}, C_{1-\frac{\alpha}{2}}\right\}$;
Return $I_\alpha(\boldsymbol{x}_{\text{test}})$

D) **Conformal inference**: A relatively recent method for calibrating PIs [47], *conformal inference* is another computational procedure that requires milder assumptions. Namely, no strict assumptions are made about $\hat{y}_N$ aside from symmetry, and the underlying distribution P of the data $\{Z_i = \left(x_{train}(i), y_{train}(i)\right) \mid i = 1, \dots, N\}$ requires the milder assumption of exchangeability rather than *i.i.d.* However, what is remarkable about this method is that it still produces PIs with valid out-of-sample coverage, see [47].
A general description of the procedure is as follows, assuming we want to make a PI for a new test-point (x_{N+1}, y_{N+1}). For every value $y \in \mathbb{R}$ of a suitable grid, a separate regression model $\hat{y}_N^y$ is trained on the augmented data-set $\{Z_1, \dots, Z_N, (x_{N+1}, y)\}$. The corresponding residuals are then:

$$R_{y,i} = |y_{train}(i) - \hat{y}_N^y(x_{train}(i))|, i = 1, \dots, N \text{ and } R_{y,N+1} = |y - \hat{y}_N^y(x_{N+1})|, \tag{4.29}$$

the latter of which is ranked among the remaining residuals to compute the proportion of points whose residuals are smaller than the last one:

$$\pi(y) = \frac{1}{N+1} \sum_{i=1}^{N+1} \mathbb{1}(R_{y,i} \le R_{y,N+1}). \tag{4.30}$$

From exchangeability and symmetry of $\hat{y}_N$, it can be seen that $\pi(y_{N+1})$ is uniformly distributed over $\{\frac{1}{N+1}, \frac{2}{N+1}, \dots, 1\}$, which implies:

$$\mathbb{P}\left((N+1)\pi(y_{N+1}) \le \lceil (1-\alpha)(N+1) \rceil\right) \ge 1-\alpha. \tag{4.31}$$

A $(1-\alpha)$-level PI can then be formed by inverting 4.31 over the selected range of y:

$$C_{conf}(x_{N+1}) = \{y \in \mathbb{R} : (N+1)\pi(y) \le \lceil (1-\alpha)(N+1) \rceil\}. \tag{4.32}$$

The above procedure produces PIs with a valid marginal coverage rate and that do not over-cover (i.e. unnecessarily wide intervals) as proven in [47], under the assumption of $\left\{(x_i, y_i) \mid i = 1, \dots, N\right\}$ being exchangeable, but it is very computationally intensive. This is due to the fact that at every new test point, many separate models have to be fitted (depending on the chosen grid for y). Instead, we mainly consider a much more efficient variant, called

split conformal inference, which is essentially a re-branded jack-knife procedure that only splits the data once, see Algorithm 3. However, this procedure retains the out-of-sample coverage property of full conformal inference, see [47].

To improve the PIs adaptiveness to the data, i.e. to have locally varying width, we use the approach discussed in [47]. The authors use a separate regression function $\hat{\rho}_N$, trained to predict the absolute training residuals, which in- or deflates the intervals depending on the input. So simply put, $\hat{\rho}_N$ can predict the error spread, and for input data where it thinks this spread is large, the PIs should correspondingly increase in length.

Note that due to its randomness, we will need to average its results over several runs to evaluate it. As a way to counter-act this randomness, we also consider a recent extension of S-CI called *multi split conformal inference* [71], which is summarized in Algorithm 4. This method aggregates separate single intervals from S-CI, by keeping those values of y that occur with a proportion $\Pi_\beta^y = \frac{1}{B}\sum_{b=1}^{B} \mathbb{1}\left\{\boldsymbol{y} \in I_\beta^{[b]}(\boldsymbol{x}_{test})\right\}$ of more than τ in B separate PIs $I_\beta^{[b]}(\boldsymbol{x}_{test})$ (with confidence level $\beta = \alpha(1-\tau)$ each). In [71], it is shown that by combining PIs in this way, the eventual multi-SCI interval $I_\alpha^\tau(\boldsymbol{x}_{test})$ will have a nominal coverage of $1-\alpha$ (assuming again of course, that the exchangeability assumption is satisfied). As usual, since we have some corresponding smoothing parameters, we need to apply cross-validation to find the best combination (though it will be restricted to a small grid for computational reasons).

Algorithm 3: Split Conformal Prediction

Data: $\boldsymbol{x}_{\text{train}}$, $\boldsymbol{y}_{\text{train}}$, $\boldsymbol{x}_{\text{test}}$, $\alpha \in (0,1)$, regression algorithms $\hat{y}_N(\cdot)$, $\hat{\rho}_N(\cdot)$
Result: $(1-\alpha)$% prediction interval $I_\alpha(\boldsymbol{x}_{\text{test}})$ for $\boldsymbol{y}_{\text{test}}$
Randomly split $\{1,\dots,N\}$ into two equal-sized subsets A_1, A_2;
$\hat{\mu} = \hat{y}_N\left(\{(x_{train}(i), y_{train}(i)) : i \in A_1\}\right)$;
$\hat{\rho} = \hat{\rho}_N\left(\{(x_{train}(i), y_{train}(i)) : i \in A_1\}\right)$;
$R_i = \frac{|y_{train}(i) - \hat{\mu}(x_{train}(i))|}{\hat{\rho}_N(x_{train}(i))}, i \in A_2$;
d = the kth smallest value in $\{R_i : i \in A_2\}$, where $k = \lceil\left(\frac{N}{2}+1\right)(1-\alpha)\rceil$;
Return $I_\alpha(\boldsymbol{x}_{test}) = \left[\hat{\mu}(\boldsymbol{x}_{test}) - \hat{\rho}_N(\boldsymbol{x}_{test})\cdot d, \hat{\mu}(\boldsymbol{x}_{test}) + \hat{\rho}_N(\boldsymbol{x}_{test})\cdot d\right]$

Algorithm 4: Multi Split Conformal Prediction

Data: $\boldsymbol{x}_{\text{train}}$, $\boldsymbol{y}_{\text{train}}$, $\boldsymbol{x}_{\text{test}}$, $\alpha \in (0,1)$, regression algorithms $(\hat{y}_N(\cdot), \hat{\rho}_N(\cdot))$, number of splits B, threshold $\tau \in \left[0, \frac{B-1}{B}\right]$, smoothing parameter $\lambda \in \mathbb{N}_0$
Result: $(1-\alpha)$% prediction interval $I_\alpha^\tau(\boldsymbol{x}_{\text{test}})$ for $\boldsymbol{y}_{\text{test}}$
for $b \leftarrow 1$ *to* B **do**
  Form two random equally sized subsets $A_1^{[b]}, A_2^{[b]}$;
  Let $\beta = \alpha(1-\tau+\frac{\lambda}{B})$;
  Compute $I_\beta^{[b]}(\boldsymbol{x}_{test})$ with level β using Algorithm 3;
Compute $\Pi_\beta^y = \frac{1}{B}\sum_{b=1}^{B} \mathbb{1}\left\{\boldsymbol{y} \in I_\beta^{[b]}(\boldsymbol{x}_{test})\right\}$ for some grid of y;
Return $I_\alpha^\tau(\boldsymbol{x}_{test}) = \left\{y \in \mathbb{R} : \Pi_\beta^y > \tau\right\}$

Results

To adequately assess how the OLS model can be used to predict energy consumption, we consider four variants LM_{base_1}, LM_{base_2}, LM_{RFE}, LM_{high}. These are defined as follows:

• LM_{base_1}: This is the ordinary base model, applied to all the variables that were available after perfectly collinear variables were removed.

• LM_{base_2}: Same model as the previous one, but it only uses the variables that were available after the previously described method for mitigating multi-collinearity was applied. Removing variables in this way could potentially also increase predictive performance, as a less complicated covariance structure might mean that its differences between the training and test data become less pronounced. Furthermore, this can also be seen as a coarse first attempt at filtering out the unwanted variables.

• LM_{RFE}: As another rudimentary attempt at filtering out the irrelevant variables before the model is fitted, this variant uses *recursive feature elimination* as described in appendix B.3. Hereby, we use the same initial variables as for LM_{base_1}.

• LM_{high}: Using whichever variant works best, we then try to assess whether including higher-order terms improves the

fit. To this end, we again use a rudimentary approach where we use variable importance measures to select the 5 most important variables, and then include all second order interactions between these. Our primary reason for using this model is to make a pre-liminary assessment for the presence of complicated patterns in the data, which a more advanced regression model should use if we hope to improve predictive performance.

The results for the whole data at once are tabulated in tables 4.1a and 4.1b, which respectively contain the results when the outliers were present and non-present, where we used the earlier described two-step procedure to detect and remove outliers. Comparing between these two main scenarios, it is clear that outliers have a very adverse effect on predictive performance, and that they should therefore be removed in order to optimize predictive performance.
Comparing the first two models, it is clear that removing all the multi-collinear variables gives worse results, likely due to too much information being discarded with such a procedure. More precisely, with a sufficiently large size for both the training and test data, covariance relations between the full list of variables can still be accurately determined and should not be structurally different (if the reasonable assumption is made that such covariance relations do not suddenly change between both periods).
The third model which employs *RFE* also does not appear to improve results, in fact it largely has identical performance as LM_{base_1}. In other words, it does not tend to filter out any of the variables. Before concluding that we should therefore use the full list of variables, we should keep in mind one of the main disadvantages of *RFE*, namely that it can not handle multi-collinearity well. When considering which variables to exclude, it uses a variable importance measure as in equation 4.33 (in the context of linear regression models) to rank the variables, and then it excludes those variables that rank last. However, 4.33 becomes small when the variance of the coefficient estimate is large, which is always the case for collinear variables regardless of their importance in predicting the output variable. Therefore it first excludes collinear variables, and if the model happens to have worse predictive performance, it will immediately jump to the conclusion that all the variables should be used.
Concerning the last model LM_{high}, it can be seen that when outliers are retained, it improves the results (if only a little), although it has much worse performance when outliers are removed. Of course, with such a large amount of variables and therefore possible interactions or other complex patterns, we should not expect an improvement when we only consider a tiny part of it. However, the fact that the improvement is visible in some way, should tell us that there is potential for improved predictions with more advanced machine learning models that can find many of such complicated patterns automatically.
The point accuracy results for every type of period are shown in table 4.2, and the specific results for the top 10 products are shown in Appendix D. Comparing the results of LM_{base_1} across the products, the first obvious conclusion is that the performance seems to depend on the product we need to make predictions for. We can also note that limiting the training data to the relevant product generally decreases performance, with the exceptions of *0750A2*, *B380* and *B337*. As for the performance of the model for new products, which consists of *B953* (since it did not appear in the training data, see figure 4.3) and of all products outside the top 10 that were not present during training, the errors are expectedly on the higher side. This is likely because a rather constrained and inflexible model like linear regression would probably not have good generalizable performance for products with never before-seen characteristics.
For the active production periods, we can see that better results are always obtained when we limit the training data to active production periods as well, rather than using the full data. Moreover, when comparing to the overall scenario, we see that while the error metrics have decreased (RMSE, MAE), the squared correlation R^2 has likewise decreased. To understand this seemingly contradictory result, if we look again at how R^2 was calculated in 4.3c (i.e. $R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$), we see that the second term is divided by a term that is proportional to the estimated standard deviation of the output. During a more stable period such as active production, the standard deviation will of course be lower, and in this case its decrease outweighs that of the average error, which causes R^2 to decrease despite the predictions improving. Misleading results like these again highlight why one should use multiple metrics.
As for the transitional periods, it seems that the model has an easier time adapting to longer transitional periods, likely because these periods are easier to distinguish than the short variants (for example, the energy consumption is notably smaller during the longer shut-down periods). So we can attribute the increased performance during these periods to similar reasons as for periods for active production, i.e. they represent long, stable periods where nothing much changes. On the other hand, the worst performance occurs during the short transition periods, likely because the model has trouble adapting to such small snippets of the data.

Predictive point accuracy (Overall and with outliers)			
Model	Metrics		
	RMSE	MAE	R^2
LM_{base_1}	118	1.49	0.1E-3
LM_{base_2}	118	1.56	0.1E-3
LM_{RFE}	118	1.49	0.1E-3
LM_{high}	**118**	**1.46**	$\mathbf{0.1E-3}$

(a)

Predictive point accuracy (Overall and without outliers)			
Model	Metrics		
	RMSE	MAE	R^2
LM_{base_1}	**1.22**	**0.85**	**0.81**
LM_{base_2}	1.35	0.92	0.78
LM_{RFE}	1.22	0.85	0.81
LM_{high}	273	3.44	0.7E-4

(b)

Table 4.1: Evaluation metrics for the four linear regression models $LM_{base_1}, LM_{base_2}, LM_{RFE}, LM_{high}$, when outliers are retained or removed. RMSE and MAE show the average error in GJ. Also, for the product IDs, "Full" means that the whole training set was used, and "Spec" means that only the specific training data with the product was used (correspondingly, an "X" means that the training data did not contain the product).

OLS: Predictive Point Accuracy							
Metrics	Period						
	Overall	Active Production		Transition			
		Test only	All	Long	Medium	Short (diff)	Short (same)
RMSE	1.22	1.2	0.74	0.62	1.08	2.23	1.85
MAE	0.85	0.92	0.57	0.37	0.75	1.5	1.44
R^2	0.81	0.17	0.36	0.37	0.6	0.02	0.18

Table 4.2: Point evaluation metrics for LM_{base_1}, where under "Active Production" we selected these periods among the test data only or the whole data.

We try to summarize the results for the top 10 products in figure 4.3, where we have also plotted the proportions of occurrences in the training and test data, as an attempt to further understand the results. We do not spot any relationship between the errors and the proportions. Also, the error for the specific datasets varies more, which makes sense as smaller samples would be more affected by these proportions.

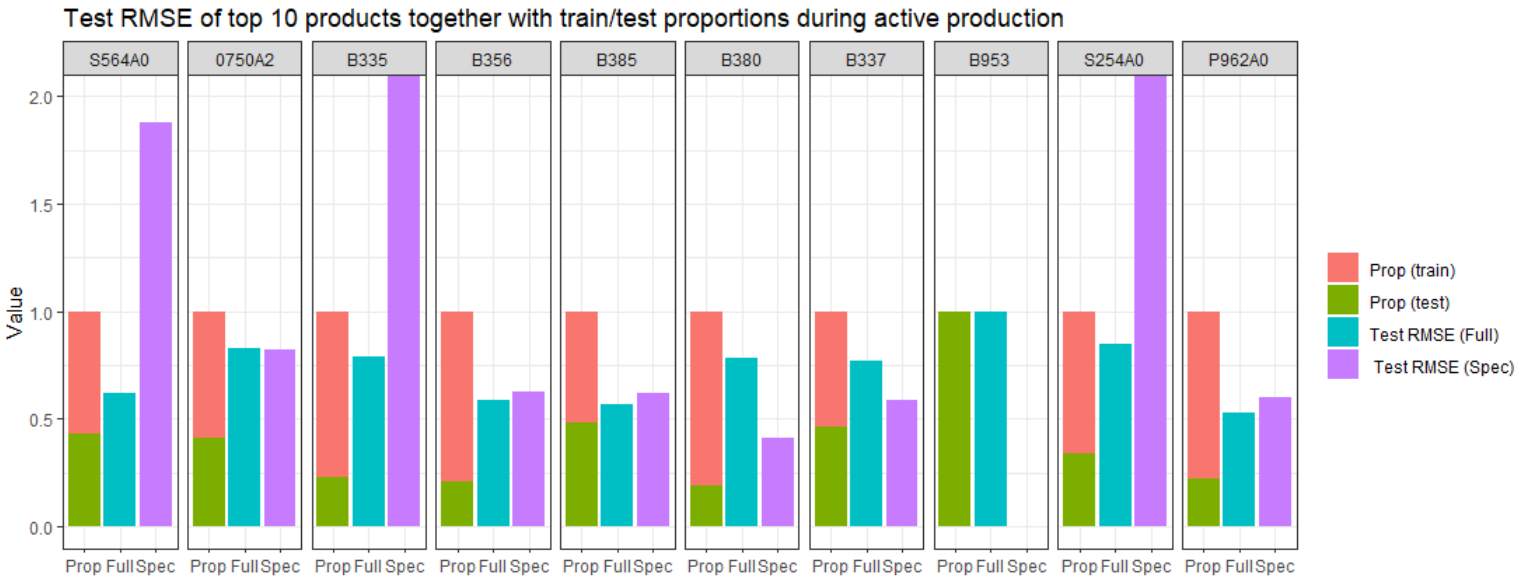
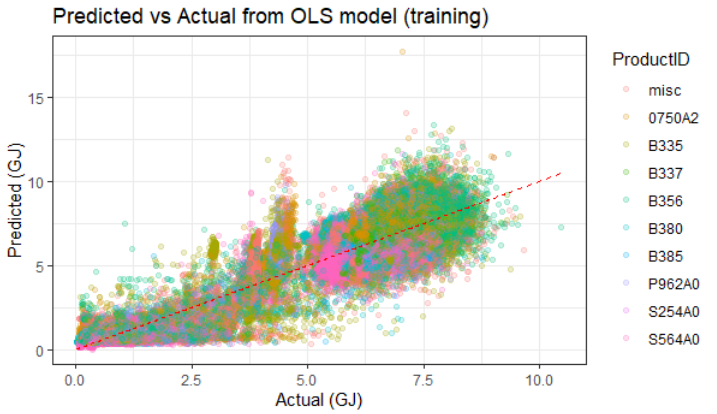


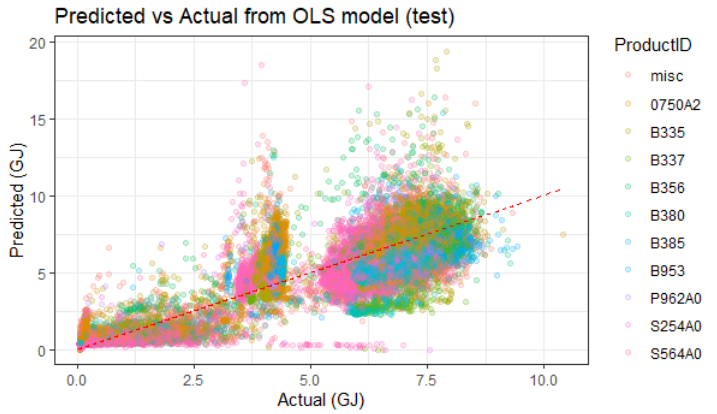
Figure 4.3: Test RMSE of LM_{base_1} for the top 10 products, both when the full and specific training datasets were used. Additionally, we show the proportions between the occurrences in the training and test periods.

In figure 4.4 we show some diagnostic plots for LM_{base_1}. Plots like these are not only meant to assess performance, but to see whether the assumptions for the specific model are adhered to (in the case of linear regression, these assumptions are described in section 4.2). We show the results for both the training as test set.
Figures 4.4a and 4.4b show the predicted values against the actual values, with the diagonal red line representing the

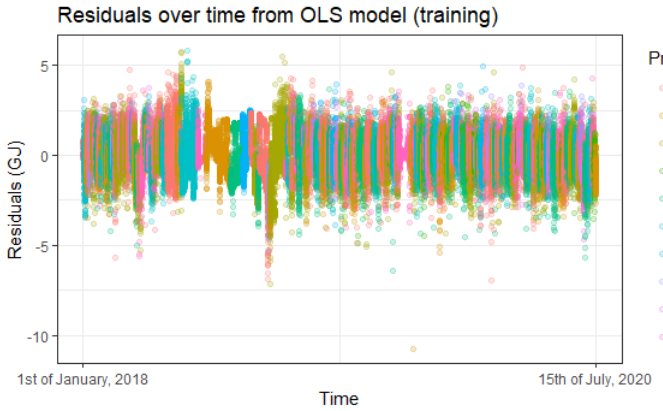
ideal situation where these values are equal and the different colours representing the top ten products. It would seem, and this is especially clear for the test data, that the predictions start to deviate the most for the middle portions of the data, indicated by the relatively large upward spikes. A preliminary explanation would be that these portions represent transitory periods between stable periods such as active production and long-term shutdown periods, and are therefore more difficult to model.
Figures 4.4c and 4.4d are used to assess homogeneity of the residual variance, i.e. do the residuals vary constantly across time? It is clear that this is not the case, as there are many instances where the variance suddenly increases or decreases, irrespective of time and product. These figures can also be used to assess when the actual values are lower than expected, as the residuals are then negative. For example, during September 2021 there was a clear down-ward trend in residuals (however, rather than indicating higher efficiency, this could also be a result of the model's poor performance during certain periods).



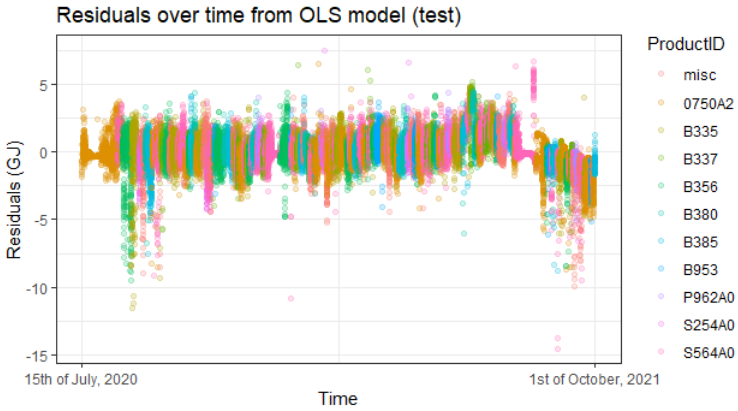
(a) Predicted values from the LM_{base_1} model plotted against actual values, for the training data.



(b) Predicted values from the LM_{base_1} model plotted against actual values, for the test data.

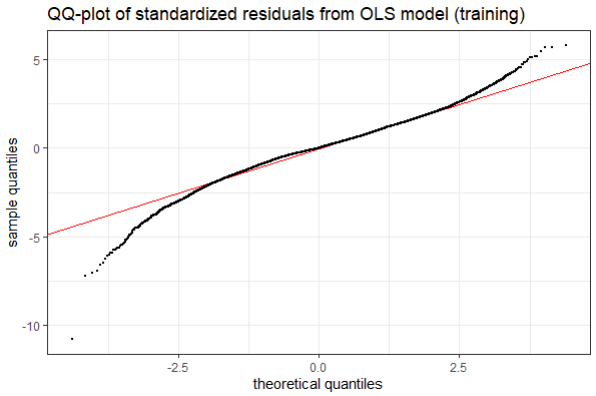


(c) Residuals over the training period for the LM_{base_1} model.

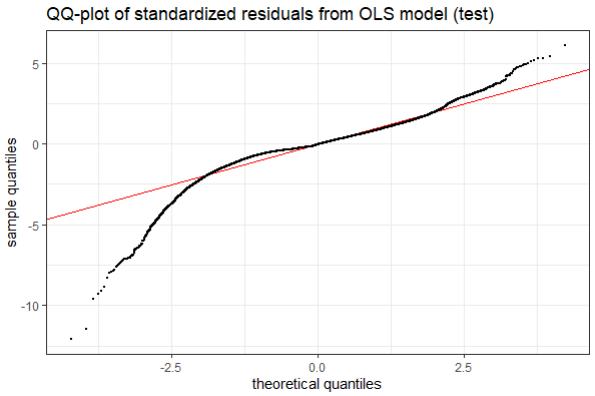


(d) Residuals over the test period for the LM_{base_1} model.

Figure 4.4: Diagnostic plots for the LM_{base_1} model. The colored points denote the top products.



(e) Sample quantiles of the residuals from the training data are compared with the theoretical quantiles from a normal distribution, to assess normality.



(f) Sample quantiles of the residuals from the test data are compared with the theoretical quantiles from a normal distribution, to assess normality.

Figure 4.4: Diagnostic plots for the LM_{base_1} model. The colored points denote the top products.

Finally, figures 4.4e and 4.4f plot the sample quantiles of the standardized residuals against the theoretical quantiles of a standard normal distribution. These are used to assess normality of the residuals, with deviations from the red line representing non-adherence to this assumption. In general, it can be seen that the residuals have heavy tails, meaning that they have more extreme values than would be expected from a normal distribution.
These diagnostic plots support the notion that the assumptions for a simple OLS model are not applicable for this dataset, and that this kind of model is therefore inappropriate to predict the energy consumption.

To assess the predictive performance in terms of intervals, we used the previously discussed methods for calibrating PIs on the LM_{base_1} model. Table 4.3 shows the results for 95% intervals, which means that the expected proportion of values falling inside the interval is 0.95.
If we first compare the analytic, bootstrap and quantile regression approaches, we see that they are not capable of providing nominal coverage. For quantile regression, this seems to be especially caused by its small width. It is also clear that this approach has better adaptability, since it still manages to have better coverage and score despite the decrease in width. Therefore, we omit the analytic and bootstrap approaches from now on.
The only methods that seems to provide a nominal coverage are S-CI and Multi S-CI, though it comes at the cost of an extremely large average width and score. In this case, the width seems to be blown up by the contributions from the regression model fitted on the absolute residuals, as visualized in figure 4.5. In turn, this likely means that the OLS model has some large residuals, which is unsurprising as the OLS model is known to be non-robust (i.e. sensitive to unusual parts of the data). Therefore, there might still be merit in applying S-CI for more robust models, as we shall do later on. Also as mentioned before, for multi S-CI we had to perform cross-validation, which we performed over a small grid for $0 \leq \tau \leq 0.9$ and $0 \leq \lambda \leq 4$.
For now, we apply quantile regression since it has the lowest score, and it is also a much more robust method due to its use of quantiles.

Predictive interval accuracy of OLS model			
Method	Metrics		
	Width	Coverage	Score
Analytic	7.79	84.3	8.58
Bootstrap	8.57	84.8	9.37
Quantile regression	**2.9**	85.4	**4.41**
S-CI	2.4E33	**95.8**	2.4E33
Multi S-CI	1.5E17	94.1	1.5E17

Table 4.3: Evaluation metrics of the five methods used to construct 95% PIs, while using the LM_{base_1} model. For S-CI we averaged the results over ten runs.

In table 4.4 we show the predictive interval results for quantile regression, where as before we consider every kind of period. Again, the active production periods show a clear improvement across the board compared with the overall scenario. This is likely due to us only considering a specific part of the process, which means that we avoid all the uncertainty that comes from transitioning between the different states of the process. Therefore, the data represents a more stable period, on which the values can be easier captured with an interval. Also, we see that neither limiting the training data

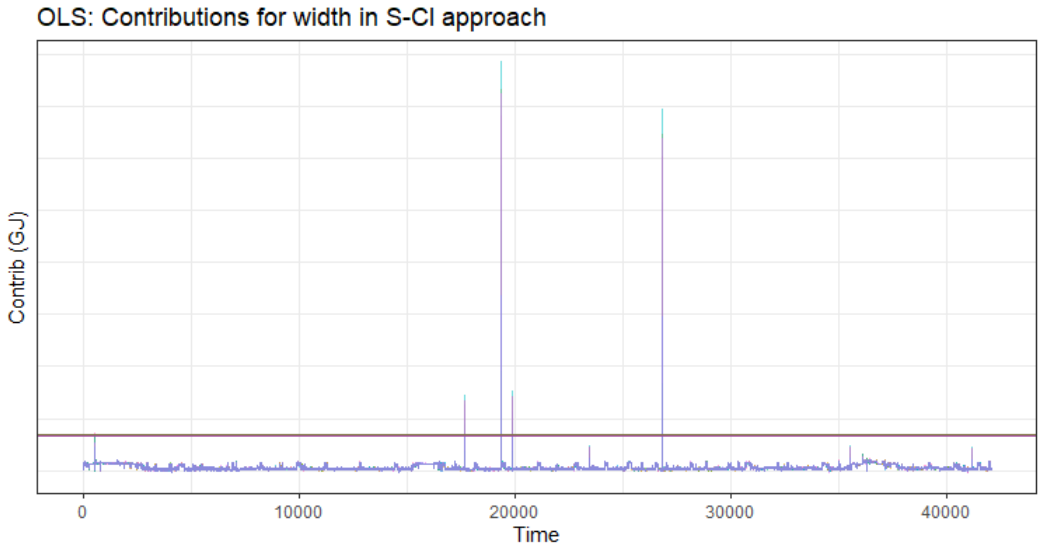


Figure 4.5: The contributions to the width of the PIs formed from S-CI, colored for 10 separate runs. The horizontal lines represent the fixed quantiles from the scaled residuals, and the non-constant lines are the predicted absolute residuals from a separate regression model (i.e. the two separate contributions to a PI's width from S-CI as discussed before).

to active production periods or specific products (see appendix D) are beneficial, especially since the coverage tends to decrease a lot. Therefore, we would recommend to use the full training data for constructing PIs. By now comparing the products, we can again conclude that the new products tend to be more difficult to model alongside *0750A2*, as they tend to have the highest scores and lowest coverages.
Also, the PIs seem to have a particularly hard time adapting to the transition periods, either in the form of low coverage (long periods) or high scores (short periods). In general, the performance is worse than for active production periods.

OLS: Predictive Interval Accuracy							
Metrics	Period						
	Overall	Active Production		Transition			
		Test only	All	Long	Medium	Short (diff)	Short (same)
Avg. Width	2.9	2.65	2.16	2.35	3.18	4.5	4.29
Coverage	85.4	91	82.8	62.5	87.1	92.9	95.2
Score	4.41	3.7	4.04	3.05	4.16	8.63	5.68

Table 4.4: Interval evaluation metrics for LM_{base_1}, for the 95% PIs formed from quantile regression. Under "Active Production" we selected these periods among the test data only or the whole data.

The question of what the important dependencies of energy consumption are as measured by a base linear regression model, would normally be assessed by the so-called *t-values* [36]. This metric is specifically used by linear regression models to measure variable importance, and is defined for a variable X_j as:

$$t_j = \frac{\hat{\beta}_j}{\text{SE}\left(\hat{\beta}_j\right)}, \tag{4.33}$$

where $\hat{\beta}_j$ was defined in equation 4.9 and $\text{SE}\left(\hat{\beta}_j\right)$ can be calculated from equation 4.12. While this is simple to calculate, it returns misleading results when there is multi-collinearity as we have alluded to before. Say that we have a predictor X_k that is strongly correlated with other predictors. Since its variance estimate will be large, its t-value will always be small. Therefore, strongly correlated predictors will always be down-weighed, regardless of their importance. This provides us with another reason to use more advanced machine learning models, as they have the capacity to accurately measure variable importance despite the presence of multi-collinearity. Therefore, we will not assess the important dependencies for the current model, and instead relegate this to future models.

4.2.2. Regularized methods

Theory

The second set of models that we consider are from the regularized family, which means that they introduce an extra penalty term in order to make the model more parsimonious as compared with the OLS model:

$$\hat{\beta}_{regul} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda(\beta) \right\}. \tag{4.34}$$

The second term therefore penalizes solutions that are too 'big', which would otherwise result in models that can easily over-fit. A nice property of regularized methods is that they can make the models simpler in a continuous fashion, rather than discrete step-wise procedures (like RFE that we considered before).
Otherwise than this, they essentially still fit the same kind of model as OLS, and therefore we do not expect big increases in performance. However, these models should provide a better picture of the outcome variable's dependencies, as they are not as strongly impacted by collinearity issues.
We now give a description of the most popular regularized variants, called ridge, lasso and elastic net. For the PIs, we use the same methods as before with the exception of the analytic and bootstrap approaches.

Ridge

Rather than selecting an optimal subset of variables through some step-wise procedure, it is more efficient to incorporate an extra penalty term when minimizing expression 4.7. In other words, the solution $\hat{\beta}_{ridge}$ is found from the following optimization problem:

$$\hat{\beta}_{ridge} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}, \tag{4.35}$$

where λ is a complexity parameter. Within statistical literature, this is often referred to as a regularized least squares problem, referring to the fact that we try to regularize the full solution set of a traditional OLS model. This regularization occurs through the added penalty term, where complex models containing more and larger coefficients are penalized more compared with simpler models. The amount of penalization is controlled by the parameter λ, so the model has to be assessed over a grid of values for this parameter in order to find the best possible model for a dataset. As variables with larger scales get penalized relatively more, especially due to the L2-norm, all the continuous variables should be standardized beforehand.
Aside from efficiency, regularized methods provide another advantage compared with step-based methods. As step-wise regression is a discrete process, the chosen set of predictors can vary wildly. In other words, the variance of models based on step-wise procedures can be very large, and therefore have poor prediction accuracy. This problem is circumvented with a continuous optimization procedure as in 4.35.

As was already obvious from expression 4.35, the penalty term does not include the intercept β_0. If the intercept is also penalized, then the model is not invariant under constant shifts of the training output variable $\boldsymbol{y}$. This means for example that when a constant c is added to $\boldsymbol{y}$, then the predicted values $\hat{\boldsymbol{y}}$ do not necessarily change with c.
Therefore, 4.35 should be rewritten slightly to account for the fact that β_0 and the other coefficients need to be estimated separately:

$$\underset{\beta_0 \in \mathbb{R}, \beta \in \mathbb{R}^p}{\operatorname{argmin}} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}. \tag{4.36}$$

Using the fact that x_j was standardized, and that therefore $\frac{\sum_{i=1}^{N} x_{ij}}{N} = 0 \Rightarrow \sum_{i=1}^{N} x_{ij} = 0$, we can easily solve for β_0 first:

$$\begin{aligned} \frac{\partial}{\partial \beta_0} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 = 0 \\ \Rightarrow \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right) = 0 \\ \Rightarrow \hat{\beta}_0 = \frac{\sum_{i=1}^{N} y_i}{N} = \bar{\boldsymbol{y}}. \end{aligned} \tag{4.37}$$

The other coefficients can be solved similarly as with OLS, keeping in mind that $\boldsymbol{X}$ is now a $p \times p$ matrix:

$$\begin{aligned} &\frac{\partial}{\partial \beta} \left(\boldsymbol{y} - \boldsymbol{X}\beta \right)^T \left(\boldsymbol{y} - \boldsymbol{X}\beta \right) + \lambda \beta^T \beta = 0 \\ &\Rightarrow \boldsymbol{X}^T (\boldsymbol{y} - \boldsymbol{X}\beta) + \lambda \beta = 0 \\ &\Rightarrow \hat{\beta}_{ridge} = \left(\boldsymbol{X}^T \boldsymbol{X} + \lambda \boldsymbol{I} \right)^{-1} \boldsymbol{X}^T \boldsymbol{y}, \end{aligned} \tag{4.38}$$

where $\boldsymbol{I}$ is the $p \times p$ identity matrix. Its expectation and variance are:

$$\begin{aligned} \mathbb{E}(\hat{\beta}_{ridge}) &= \left(\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{X}^T\boldsymbol{X}\beta \\ \text{Var}(\hat{\beta}_{ridge}) &= \sigma^2 \left(\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{X}^T\boldsymbol{X} \left(\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I}\right)^{-1}, \end{aligned} \tag{4.39}$$

so for larger values of λ, the bias increases and the variance decreases.

A potential advantage of ridge regression is that it can naturally deal with multi-collinearity. When $\lambda > 0$, the estimate in 4.38 is always defined, i.e. $\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I}$ is always non-singular. When we look at the variance of the OLS estimator in 4.12, it is obvious that multi-collinearity will cause it to blow up, on account of $\boldsymbol{X}^T\boldsymbol{X}$ becoming singular. By preventing the matrix from becoming singular, we can mitigate the effect of multi-collinearity. This is also referred to as the *joint shrinkage* property of ridge regression, which means that it simultaneously shrinks the collinear variables towards zero by an equal amount (a proof is given in appendix B.4). Eventually, the underlying estimates become more stable.
Essentially, the hope is that the increase of bias is off-set by a larger decrease in variance (the prediction accuracy as measured by MSE is the sum of squared bias and variance, see [36]).

To quantify the complexity of a ridge model, we use the *effective degrees of freedom* which takes as input the penalty value. Let $\boldsymbol{H}_{\lambda}^{ridge} = \left(\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{X}^T$ denote the "hat" matrix, which contains all the information needed to form the estimator $\hat{\beta}_{ridge}$. The effective degrees of freedom is then calculated by taking its trace:

$$\begin{aligned} df(\lambda) &= \text{tr}\left(\boldsymbol{H}_{\lambda}^{ridge}\right) \\ &= \sum_{j=1}^{p} \frac{d_j^2}{d_j^2 + \lambda}, \end{aligned} \tag{4.40}$$

where d_j is a singular value of $\boldsymbol{X}$. For the derivation we refer to [36].
Finally, we would just like to mention that most of the assumptions for ridge regression are identical to OLS, with the exception of multi-collinearity (as it is handled automatically) and normality of the residuals.

Lasso
Like ridge regression, the lasso model regularizes the OLS optimization problem, but it uses a different penalty term:

$$\hat{\beta}_{lasso} = \underset{\beta_0 \in \mathbb{R}, \beta \in \mathbb{R}^p}{\text{argmin}} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}. \tag{4.41}$$

Solving for β_0 is identical as for ridge regression, so we omit that and focus on the other parameters β. Using a L1-norm may seem like a small change, but it actually has big consequences for the obtained solution as we will explain.
Firstly, unlike with previous models, the problem 4.41 has now lost its differentiability. This is due to the fact that the absolute function $g(\cdot) = |\cdot|$ is not differentiable at zero. Therefore, it is generally not possible to analytically solve for β, unless we assume that $\boldsymbol{X}$ is orthonormal [36].
To show this, let $\boldsymbol{X}$ be orthonormal such that $\boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{I} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}$. Noting that the OLS estimator becomes $\hat{\beta} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y} = \boldsymbol{X}^T\boldsymbol{y}$, the lasso loss function 4.41 can be reformulated as:

$$\begin{aligned} \min_{\beta} \|\boldsymbol{y} - \boldsymbol{X}\beta\|_2^2 + \lambda\|\beta\|_1 &= \min_{\beta} \boldsymbol{y}^T\boldsymbol{y} - \boldsymbol{y}^T\boldsymbol{X}\beta - \beta^T\boldsymbol{X}^T\boldsymbol{y} + \beta^T\boldsymbol{X}^T\boldsymbol{X}\beta + \lambda \sum_{j=1}^{p} |\beta_j| \\ &\propto \min_{\beta} -\hat{\beta}^T\beta - \beta^T\hat{\beta} + \beta^T\beta + \lambda \sum_{j=1}^{p} |\beta_j| \\ &= \min_{\beta_1,\ldots,\beta_p} \sum_{j=1}^{p} \left(-2\hat{\beta}\beta_j + \beta_j^2 + \lambda |\beta_j| \right) \\ &= \sum_{j=1}^{p} \left(\min_{\beta_j} -2\hat{\beta}_j\beta_j + \beta_j^2 + \lambda |\beta_j| \right). \end{aligned} \tag{4.42}$$

Therefore, each parameter can be solved separately with its solution depending on the sign of β_j:

$$\left(\min_{\beta_j} -2\hat{\beta}_j\beta_j + \beta_j^2 + \lambda |\beta_j| \right) = \begin{cases} \left(\min_{\beta_j} -2\hat{\beta}_j\beta_j + \beta_j^2 + \lambda\beta_j\right) & \text{if } \beta_j > 0, \\ \left(\min_{\beta_j} -2\hat{\beta}_j\beta_j + \beta_j^2 - \lambda\beta_j\right) & \text{if } \beta_j < 0, \end{cases} \tag{4.43}$$

which leads to the solution

$$\hat{\beta}_j^{lasso}(\lambda) = \begin{cases} \hat{\beta}_j - \frac{1}{2}\lambda & \text{if } \hat{\beta}_j > \frac{1}{2}\lambda, \\ \hat{\beta}_j + \frac{1}{2}\lambda & \text{if } \hat{\beta}_j < -\frac{1}{2}\lambda, \\ 0 & \text{otherwise.} \end{cases} \tag{4.44}$$

The j-th coefficient of the lasso solution can then be summarized as follows (called the *soft-thresholding* operator, see [36]):

$$\hat{\beta}_j^{lasso}(\lambda) = \text{sign}(\hat{\beta}_j)\left(\mid \hat{\beta}_j \mid -\frac{1}{2}\lambda\right)_+, \tag{4.45}$$

where $(\cdot)_+ = \max\{\cdot, 0\}$.

To solve general lasso problems of the form 4.41, the previously proven property is often exploited. In the popular R package *glmnet* [31] for example, the corresponding *coordinate descent* algorithm iteratively optimizes for each variable separately by isolating it and using its column in $\boldsymbol{X}$ as the new orthonormal design "matrix":

$$\begin{aligned} \underset{\beta_j}{\text{argmin}} \|\boldsymbol{y} - \boldsymbol{X}\beta\|_2^2 + \lambda\|\beta\|_1 &= \underset{\beta_j}{\text{argmin}} \|\boldsymbol{y} - \boldsymbol{X}_{*,k\neq j}\beta_{k\neq j} - \boldsymbol{X}_{*,j}\beta_j\|_2^2 + \lambda \mid \beta_j \mid \\ &= \underset{\beta_j}{\text{argmin}} \|\tilde{\boldsymbol{y}} - \boldsymbol{X}_{*,j}\beta_j\|_2^2 + \lambda \mid \beta_j \mid, \end{aligned} \tag{4.46}$$

where $\tilde{\boldsymbol{y}} = \boldsymbol{y} - \boldsymbol{X}_{*,k\neq j}\beta_{k\neq j}$ and $\boldsymbol{X}_{*,k\neq j}$ denotes the entirety of $\boldsymbol{X}$ aside from its j-th column. The corresponding solution is thus found by applying 4.45, while deriving $\hat{\beta}$ from $\tilde{\boldsymbol{y}}$ and $\boldsymbol{X}_{*,j}$. This procedure is repeated for each variable in turn, until no further updates cause 4.41 to change beyond some threshold. Again, the model is optimized over a grid of λ to minimize the CV-estimate of the test error.

The solution 4.45 also provides some insight into lasso's unique feature, as it is clear that it can truncate the OLS estimate $\hat{\beta}$ to zero once it is below some threshold. In other words, lasso regression is capable of performing feature selection automatically. Meanwhile, ridge regression simply shrinks coefficients to very small values without being able to explicitly set them to zero (and therefore excluding them from the model). This difference can be better explained from a geometrical viewpoint, which is visualized in figure 4.6.
Solving the penalized optimization problems 4.35 and 4.41 is akin to placing a constraint on β, which for lasso and ridge are respectively $\sum_{j=1}^{p} \mid \beta_j \mid \leq t(\lambda_{lasso})$ and $\sum_{j=1}^{p} \beta_j^2 \leq t(\lambda_{ridge})$. Therefore, the corresponding constraint regions are diamond and disc shaped, which need to be intersected with the MSE contours in order to find a solution. For ridge regression, this intersection can occur anywhere on the disc, while for lasso regression this intersection usually occurs on the sharp corners. The latter happen to coincide with the axes, and therefore lasso regression is able to set coefficients to zero.
Finally, we just mention that the assumptions for lasso are the same as for OLS, except for normality of the residuals.

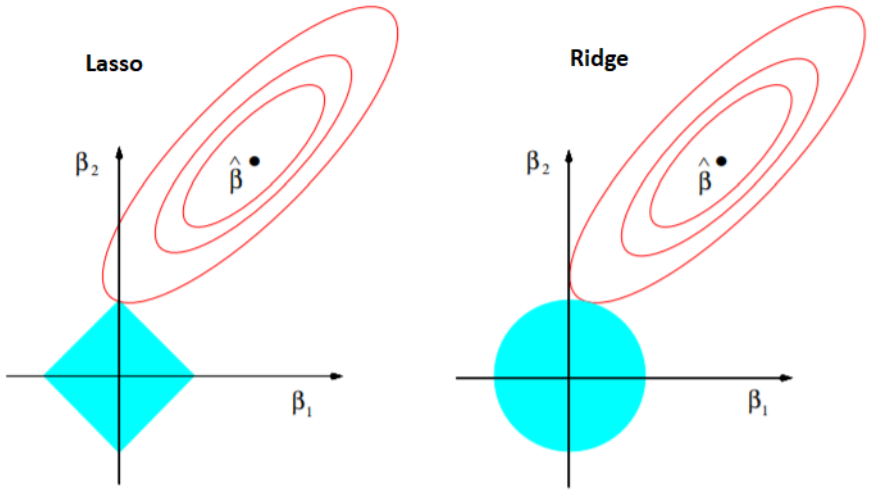


Figure 4.6: Geometrical interpretation of lasso and ridge regression. The red elliptical contours represent the level sets from the MSE error $\|\boldsymbol{y} - \boldsymbol{X}\beta\|_2^2$, and the blue regions represent the constraint regions. Figure is taken from [36] (p.71).

Elastic Net

We have seen that ridge and lasso regression have distinct advantages. Whereas ridge regression can handle multi-collinearity due to its joint shrinkage property, lasso regression can perform feature selection on its own. The question then arises, whether these two models can be combined in some way, such that both advantages can be utilized and the weaknesses mitigated. It turns out that this can be done by simply combining the respective penalty terms, which is called *elastic net* regression [36].

The new optimization problem then becomes (omitting the intercept for similar reasons as before):

$$\hat{\beta}_{elas} = \underset{\beta \in \mathbb{R}^p}{\operatorname{argmin}} \|\boldsymbol{y} - \boldsymbol{X}\beta\|_2^2 + \lambda_1 \|\beta\|_1 + \frac{1}{2}\lambda_2 \|\beta\|_2^2, \tag{4.47}$$

so it is clear that elastic net regression is some generalization of both lasso and ridge regression, with their respective strengths controlled by λ_1 and λ_2. Visually, the constraint region would be a compromise between the diamond and disc regions in figure 4.6, with its volume controlled by the penalty sizes and its shape by the relative sizes between them.

To solve 4.47, it can be rewritten into a lasso problem through data augmentation. Let $\tilde{\boldsymbol{y}} = \left(\boldsymbol{y}^T, \boldsymbol{0}_p^T\right)^T$, with $\boldsymbol{0}_p$ denoting the p-dimensional zero vector, and $\tilde{\boldsymbol{X}} = \left(\boldsymbol{X}^T, \sqrt{\lambda_2}\boldsymbol{I}_{pp}\right)^T$. We then have:

$$\|\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{X}}\beta\|_2^2 = \|\boldsymbol{y} - \boldsymbol{X}\beta\|_2^2 + \lambda_2 \|\beta\|_2^2, \tag{4.48}$$

which means that 4.47 is equivalent to:

$$\hat{\beta}_{elas} = \underset{\beta \in \mathbb{R}^p}{\operatorname{argmin}} \|\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{X}}\beta\|_2^2 + \lambda_1 \|\beta\|_1. \tag{4.49}$$

Therefore, similar algorithms as for lasso can be employed to solve 4.47. However, there is an issue related to the fact that two penalty parameters need to be optimized for simultaneously. Namely, the problem becomes indeterministic, as there can be a wide range of different combinations (λ_1, λ_2) which give similar sizes for β (and therefore similar performance). From an interpretation perspective this can also complicate things, as different solutions with different sparsity patterns may give similar performance.
To remedy this, an alternative representation of the elastic net penalty is given, using a mixing parameter α [36]:

$$\lambda \left[\alpha \|\beta\|_1 + \frac{1}{2}(1-\alpha)\|\beta\|_2^2 \right]. \tag{4.50}$$

The parameter α fixes the ratio between both penalty terms, and needs to be tuned alongside λ (this approach is also used by *glmnet*).

Results

For the previous set of models, we had attempted some rudimentary first attempts at filtering out the irrelevant variables, namely by removing the input variables that are correlated with each other and *RFE*. However, we saw that they did not improve the results, and in fact often gave back worse results. A likely explanation for this is that they are very coarse methods. In other words, they can only keep a variable as it is, or completely remove it.
What might be more beneficial, and this is indeed what the regularized models were designed to do, is to make this variable selection process more flexible. Completely removing variables means that inevitably information is discarded, so to prevent this we might instead lower their influence. Therefore, we try to retain the information that these variables contain, while avoiding over-fitting due to the amount of degrees of freedom increasing too much (i.e. the regularized variant called *effective* degrees of freedom, see 4.40). We consider three regularized models: *Ridge*, *Lasso* and *Elastic Net*. As mentioned before, elastic net requires two hyper-parameters λ and α to be estimated. To find an optimal value for the latter, we show the test RMSE for a grid of α in table 4.5 when the full data is used and λ is estimated in the same way as the other models, and conclude that $\alpha = 0.4$ gives optimal results.

Performance of elastic net for different values of α									
α	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
RMSE	1.1702	1.1699	1.1726	1.168	1.1717	1.1729	1.1712	1.1733	1.1758

Table 4.5: Elastic net evaluated in terms of RMSE for different values of α, when the full training and test data are used.

Proceeding in the same way as before, we tabulate the point accuracy of the regularized models in table 4.6. For brevity we ignore the configuration with outliers, as we have shown already that they give worse results.
If we compare the models, we can first see that ridge gives by far the worst results. As for an explanation, we must look at ridge's joint shrinkage property (see appendix B.4). The issue is that ridge regression will indiscriminately shrink all strongly correlated input variables by approximately the same amount, even if some of those variables are important for the outcome variable. On the other hand, lasso regression can more carefully pick amongst these sets of correlated variables, due to its feature selection property. Therefore, for data-sets with many correlated input variables such as ours, lasso is less likely to erroneously discard information than ridge.
Furthermore, lasso and elastic net seem to produce near identical results. As lasso is easier to fit and work with, we therefore discard elastic net. Comparing the lasso with the unregularized models from before, we see that it has slightly better results compared with the LM_{base_1} model from table 4.1b, indicating that carefully removing some variables can provide some benefit.
For the lasso model, the point accuracy for every period is shown in 4.7. Another similar finding as before is that we should likewise limit the training data to active production periods, however it seems that further limiting it to specific products (see appendix D) gives better results for more products, namely for *0750A2*, *B356*, *B385*, *B380* and *B337*. Again, this can be explained by lasso's variable selection property, as these products would likely depend on less variables during active production (i.e. they require less information during stable periods to be modelled accurately). The results for transitional periods are largely the same as for the OLS model, so we will not discuss those.
As for the top 10 products, which we visualized in figure 4.9 for clarity, it is clear that the new products again have the largest errors. Also, in terms of limiting the training data, it was clear that limiting it to specific products again gave generally worse and unreliable results. There were however some exceptions with products *B335*, *B356* and *B337*, the former two of which also formed the exceptions for LM_{base_1}. This seems to provide stronger evidence that these products have some pronounced specific patterns that can be exploited.

Predictive performance of regularized models (without outliers)			
Model	Metrics		
	RMSE	MAE	R^2
Ridge	5E52	2E50	1E-5
Lasso	**1.17**	**0.81**	**0.82**
Elastic Net	1.17	0.81	0.82

Table 4.6: Evaluation metrics for three regularized models, when outliers are removed. For elastic net we used $\alpha = 0.4$. As the models are fitted with cross-validation, the results were averaged over 10 runs.

LASSO: Predictive Point Accuracy							
Metrics	Period						
	Overall	Active Production		Transition			
		Test only	All	Long	Medium	Short (diff)	Short (same)
RMSE	1.17	1.13	0.69	0.61	1.07	2.24	1.92
MAE	0.81	0.84	0.52	0.37	0.74	1.48	1.49
R^2	0.82	0.22	0.38	0.39	0.6	0.04	0.2

Table 4.7: Point evaluation metrics for lasso, where under "Active Production" we selected these periods among the test data only or the whole data.

Tables 4.8 and 4.9 show the results for the prediction intervals, which support some of the same conclusions that we drew from the OLS model, namely that we should use S-CI for best nominal coverage, that for making predictions during periods of active production we should use the full training data to obtain best results, and from figure 4.9 it is clear that again the new products together with *0750A2* are most difficult to predict. We also see that unlike before, S-CI performs much better, with a width and score that are now not blown up by large residuals. Also, the long transitions can be captured with nominal coverage, which is a large improvement compared with the 62.5% coverage we saw with OLS.
However, despite the increase in average width, the coverage and score for the short transitions have become worse. This decrease in adaptability likely stems from the randomness of S-CI, as the performance on small parts of the data (like short transition periods) would be more susceptible to random fits. Regardless, even with Multi S-CI which is supposed to mitigate this, we only see a small universal improvement for product switches. What is also noteworthy is that it clearly performs better during active production. Overall it also provides a slightly lower width and score, at the cost of a slightly smaller coverage. Therefore, since S-CI provides the best nominal coverage overall, we will use S-CI instead of Multi S-CI.
Further comparing the interval results for the overall situation with the previous ones from OLS in table 4.3, it is clear that LASSO improves results in every way like it did with the point predictions. We also see that in general, S-CI may have a

larger width and score compared with quantile regression, but so far it is the only method capable of providing nominal coverage. At this point we therefore recommend that the lasso model, together with the S-CI approach if PIs need to be constructed, should be used to make predictive inferences.

Predictive interval accuracy of LASSO model			
Method	Metrics		
	Width	Coverage	Score
Quantile regression	**2.87**	89.6	**3.97**
S-CI	5.57	**94.9**	7.18
Multi S-CI	5.15	94.1	6.62

Table 4.8: Evaluation metrics of the constructed 95% PIs from the Lasso model. For S-CI we averaged the results over ten runs as before.

LASSO: Predictive Interval Accuracy								
Method	Metrics	Period						
		Overall	Active Production		Transition			
			Test only	All	Long	Medium	Short (diff)	Short (same)
S-CI	Avg. Width	5.57	5.61	1E12	3.71	5.59	9.57	9.24
	Coverage	**94.9**	96.7	93.8	**94.6**	**91.6**	90.5	**93.4**
	Score	7.18	6.28	1E12	3.83	7.03	18	11.4
Multi S-CI	Avg. Width	**5.15**	**5.34**	**3.57**	**2.99**	**4.63**	**9.09**	**8.54**
	Coverage	94.1	**96.9**	**95.9**	88.5	90.2	**91.3**	92.3
	Score	**6.62**	**6**	**4.34**	**3.16**	**6.11**	**17.1**	**11**

Table 4.9: Interval evaluation metrics for lasso, for 95% PIs constructed from single S-CI (averaged over 10 runs) and multi S-CI.

We visualize the predictive results in figures 4.7, 4.8 and 4.9. Figure 4.7 shows how the prediction interval looks like, for the last few months of the testing period. For the rest of the testing period, we show some additional figures in appendix A.3. Of note are the large upwards spikes that keep occurring over the entire range of the test-period, which is the likely cause of the large average width and score of S-CI that we saw before. Furthermore, there are also certain periods where the actual values (black dots) are very often above or below the point predictions (dashed line), which respectively occurs for example during June and September of 2021. This might indicate that some improvement in energy efficiency has occurred during these periods, although this would first need to be confirmed with more accurate models before a more definitive conclusion can be taken.

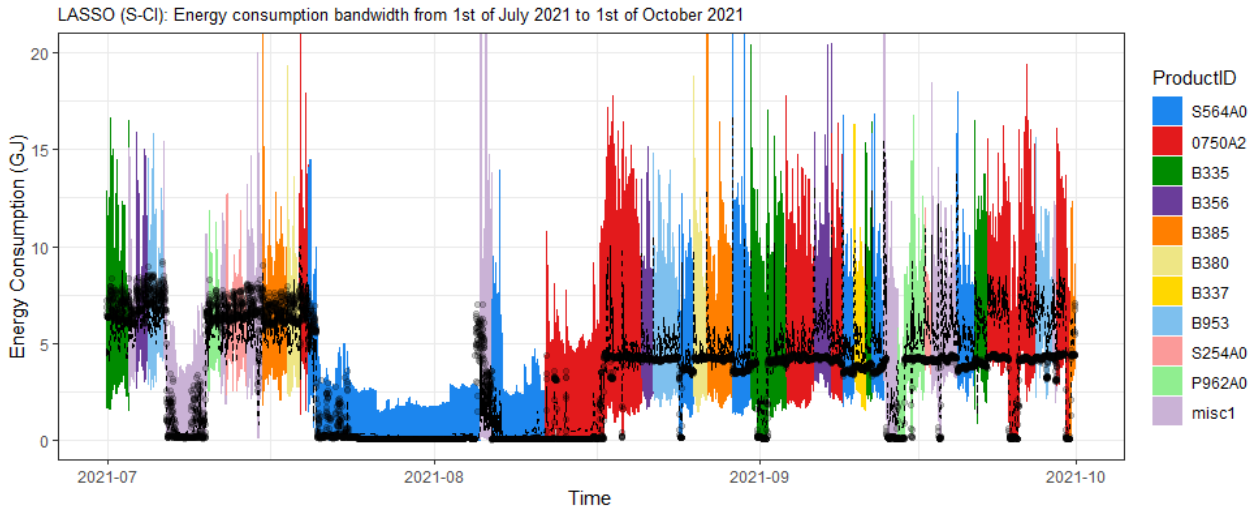


Figure 4.7: A recent snippet of the 95%-PI, colored by the top 10 products. The actual observations are denoted with the black dots, and the prediction points with the dashed line. "Misc1" denotes all other products outside the top 10.

How the predictive performance depends on time of occurrence, is summarized in figure 4.8. For each month, we show the performance for both the training and test data. This seems to clearly confirm that the predictive performance

becomes worse during June, August and September, even though the training performance does not indicate this. In particular, September is a consistent problem area, with the model then having worse point and interval performance for both the overall and active production data. For June and August this is a bit more constrained, as for example June only poses a problem for overall point predictions.

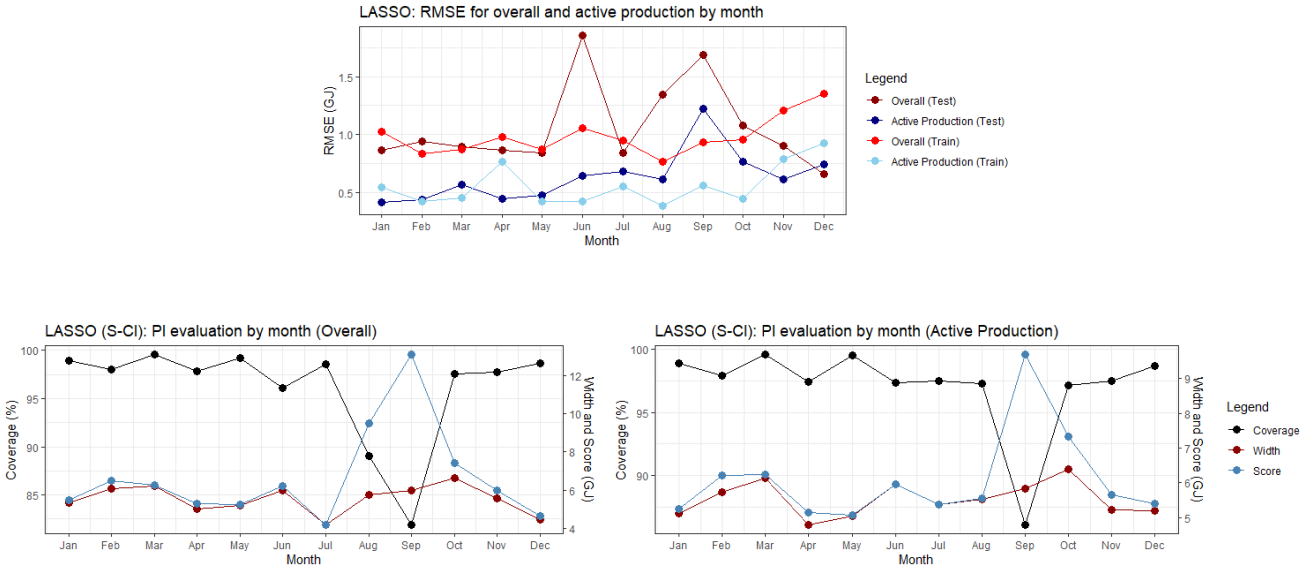


Figure 4.8: Predictive performance of the LASSO model, plotted by month for prediction points and intervals.

Figure 4.9 likewise summarizes the predictive performance, where we instead consider both the specific products as well as groups that are sorted in ascending order of the energy consumption. We see that the new products clearly perform worst as noted before. On the other hand, *P962A0* show the best results for the point predictions and PIs alike, with it respectively having the lowest test RMSE and score.
Concerning the ordered groups, the predictions seem to worsen for larger consumption values, again shown by the higher RMSE/scores. In particular, the second ordered group has the worst performance, and we hypothesize that this is caused due to it representing some kind of short-term transition period between stable periods (shut-down and active production), which we have shown to be most difficult to predict for using the current model.

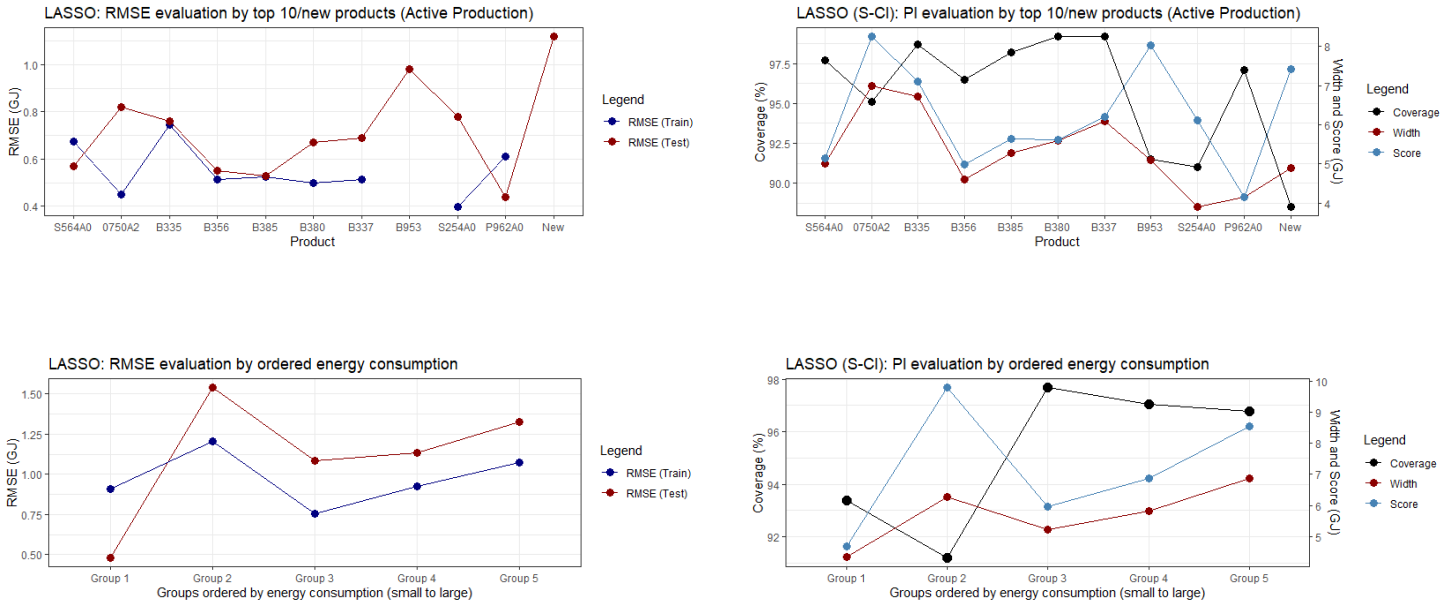


Figure 4.9: Predictive performance of the LASSO model, shown separately for the top 10 products and ordered energy-consumption groups.

Now that we have assessed the predictive performance, it is time to apply this model in order to make a first attempt at answering the main research questions. First, we need to show when the largest changes in efficiency takes place, which we can do by considering the relative contribution of over- and underestimations. For point predictions, we simply consider the relevant parts of these from the MAE metric, while for PIs we can consider the relevant part of the score function (see 4.4c). Note that the latter only considers more extreme deviations since it requires the predictions to fall outside the PI, so it provides a more conservative assessment. In figures 4.10 and 4.11 we show the error decomposition for the point predictions and PIs, where we consider the top 10 products and every month.
It can be seen that for the point predictions, the contributions from underestimations tend to be larger during active production, whereas for the overall situation this appears to be more balanced. Notable exceptions are June and September, where the latter month still shows the same imbalance for the conservative PI assessment. Therefore, in general these results seem to indicate that the efficiency has somehow decreased during periods of active production, and that any increases in efficiency would have likely occurred outside of these periods, in particular during transitory periods. It should again be noted that large contributions as visible here can be misleading, since a part of these contributions can be caused by the model simply not performing well in certain periods. For this reason, a bigger focus should be placed on active production, as the predictive model performs better for stable periods (and therefore the imbalances in under/over-estimations are more likely to stem from structural changes in the production process).
Regardless, we can not currently make any definitive statements, considering that we have as of yet only considered one model, which by itself may not be very accurate to begin with. This provides another motivation why we will consider more advanced models in the next section, so that we may obtain a better energy-baseline to compare the actual values against.

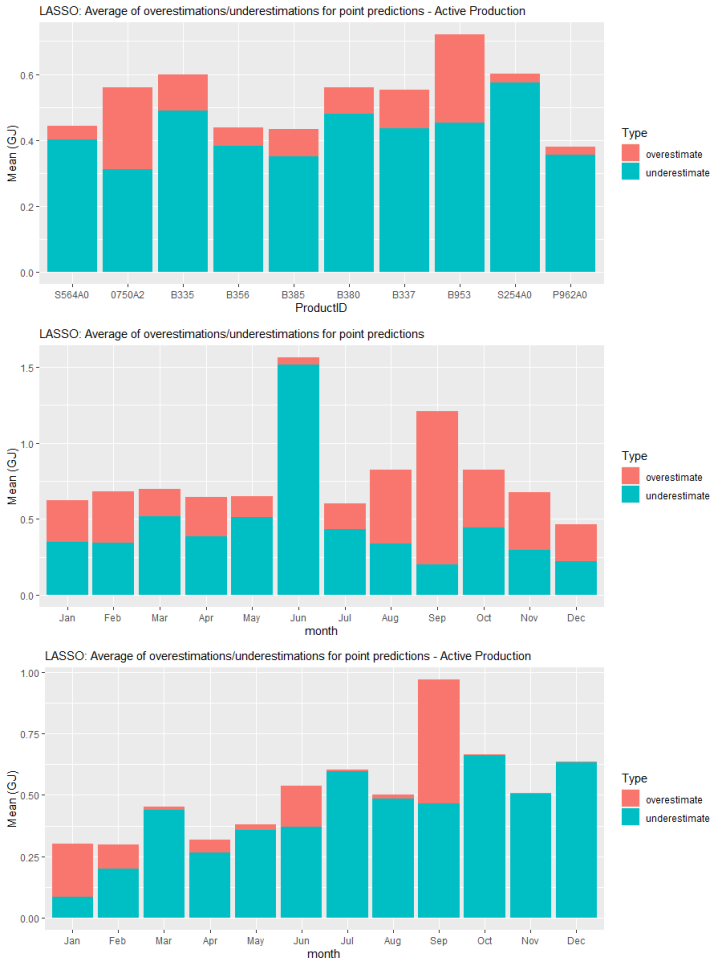


Figure 4.10: The averaged over-and underestimations for the top 10 products and 12 months.

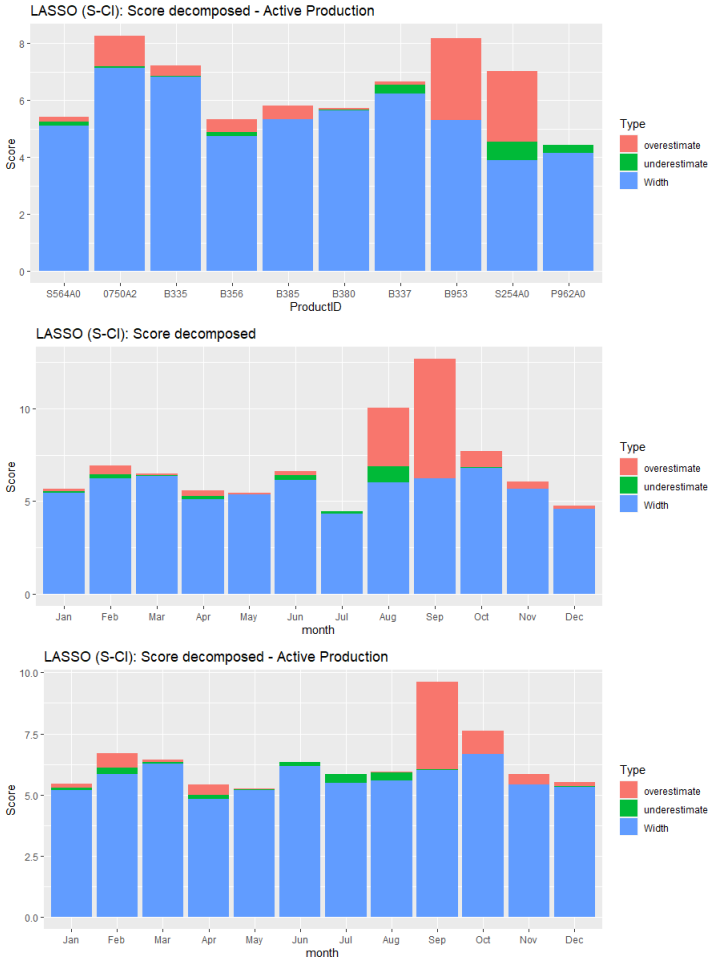
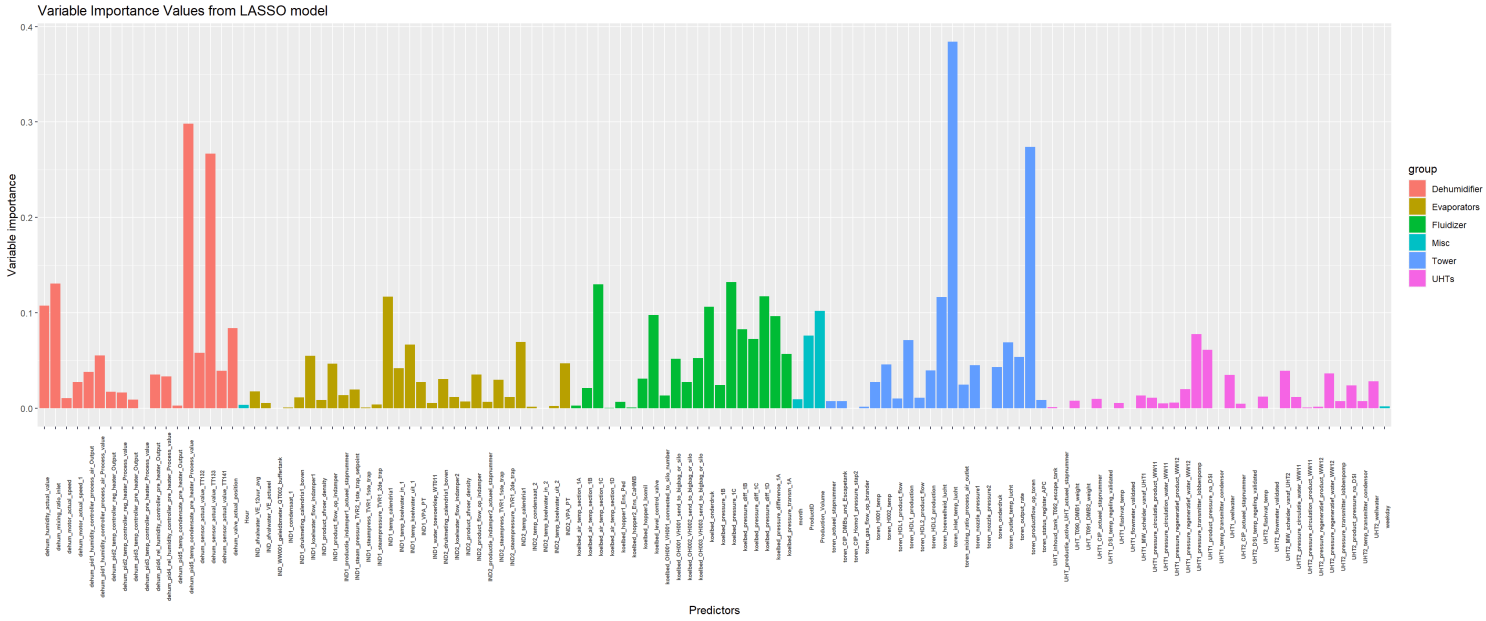


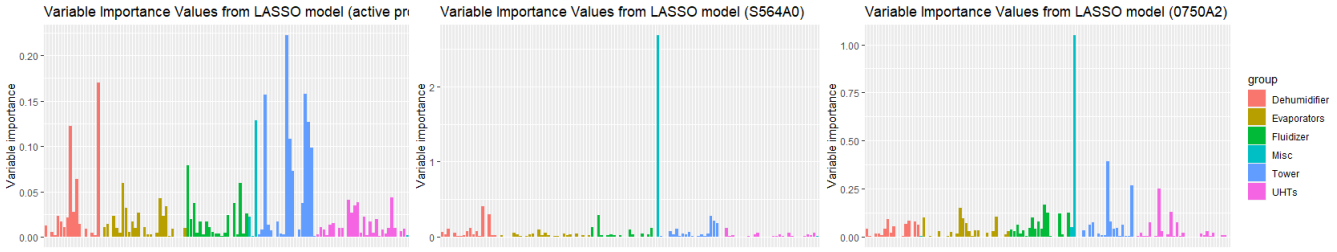
Figure 4.11: The decomposed score evaluation metric for the top 10 products and 12 months, showing the contributions from average width, over- and underestimations.

Finally, we visualize the dependencies/variable importances in figure 4.12. The importance values themselves are simply the absolute values of the standardized coefficients, while the LASSO models that produced these values were fitted with standardized variables as before, but also without the intercept term.
For the overall picture, we see the largest importances originate from the tower and dehumidifier. During active produc-

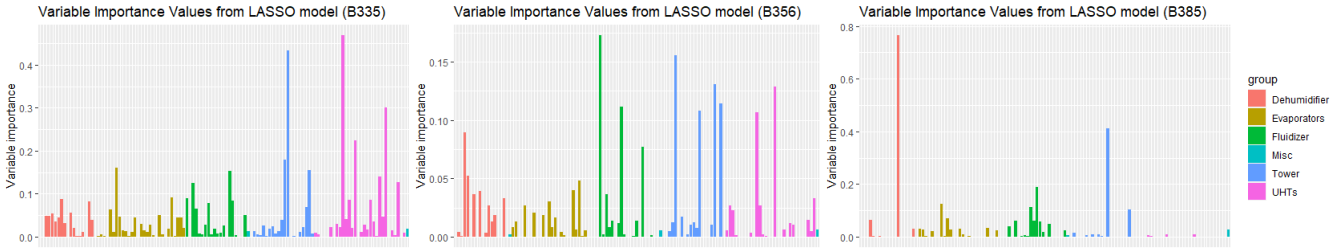
tion, the importance of the tower has clearly increased, with more of its variables taking large importances.
For the specific products, it is clear that each one has a distinct set of importance values, which indicates that the dependencies differ by product. To get a more definitive grasp of which variables are most important, we will repeat this assessment for other models later on and try to find common patterns.



(a) Full



(b) Active Production (c) S564A0 (d) 0750A2



(e) B335 (f) B356 (g) B385

Figure 4.12: Variable importance values from the LASSO model, colored by the different parts of the drying process. In (a) the full training data was used. Training data which only contain active production periods and top 5 products are shown in (b), (c), (d), (e), (f) and (g).

4.2.3. Dimension reduction methods

Theory

Like the regularized methods, the dimension reduction methods (also knows as derived input methods) try to reduce the complexity of the models. We saw with regularization that the optimization problem is modified by also taking into account the size of β, which makes sure that the complexity is controlled in a continuous and flexible manner. However, strict dimensionality reduction is usually only possible through the lasso, which simply discards the less relevant variables. An alternative way of reducing complexity is to combine the original predictors $\{X_i \mid i = 1,\dots,p\}$ such that a smaller set of orthogonal (and therefore uncorrelated) *derived* variables $\{Z_m \mid m = 1,\dots,M\}$ is obtained, where $M < p$. Each Z_m is some linear combination of the original p variables:

$$Z_m = \sum_{j=1}^{p} \phi_{jm} X_j, \tag{4.51}$$

for some *loadings* $\phi_{1m},\dots,\phi_{pm}$. In other words, we form a new data matrix $T = XP$, where the i-th column in T is a *score vector* $z_i = \{z_{1i},\dots,z_{Ni}\}$ and P is the loadings matrix. With these derived variables, we then fit an OLS model for $\boldsymbol{y} = T\beta_M$ as usual, and therefore the underlying assumptions remain strict. The final step is to transform β_M back into the scale of the original variables.

Each dimension reduction method calculates the constants $\phi_{\cdot m}, m = 1,\dots M$ in a distinct manner, and we consider two such popular methods: *Principal Component Regression* (PCR) and *Partial Least Squares* (PLS) [36]. Essentially, these models try to find directions that have maximum variance within $\boldsymbol{X}$ (PCR) or simultaneously try to maximize this variance and the covariance with $\boldsymbol{y}$ (PLS). Therefore, we would expect PLS to have better predictive performance and therefore lower bias. However, as noted in [39] (p.241), there is also potential for higher variance, which means that the MSE might not necessarily decrease. For this reason, we consider both of these methods.

Principal Component Regression

The derived variables for this model are the principal components (PCs), which are defined by the loadings that maximize their variance within $\boldsymbol{X}$ (for $Z_j, j \geq 2$, they also have to be orthogonal to previous PCs). The corresponding optimization problem for the m-th PC direction v_m is:

$$\begin{aligned} \max_{\alpha} \quad & \mathrm{Var}(\boldsymbol{X}\alpha) \\ \text{subject to} \quad & \|\alpha\| = 1, \\ & (\boldsymbol{X}\alpha)^T \boldsymbol{X} v_l = 0, l = 1,\dots,m-1. \end{aligned} \tag{4.52}$$

There is an efficient procedure for estimating the PCs (assuming that $\boldsymbol{X}$ has been standardized) [36], which is as follows:

1. Decompose $\boldsymbol{X}$ using the singular value decomposition (SVD): $\boldsymbol{X} = U\Delta V^T$, where $\Delta_{p\times p} = \mathrm{diag}\left[\delta_1,\dots,\delta_p\right]$ contains the singular values and $U_{n\times p}, V_{p\times p}$ contain the orthonormal sets of left and right singular vectors.
2. The SVD can be used to compute the spectral decomposition $V\Lambda V^T$ of $\boldsymbol{X}^T\boldsymbol{X}$, where $\Lambda_{p\times p} = \Delta^2 = \mathrm{diag}\left[\lambda_1,\dots,\lambda_p\right]$ contains the ordered eigenvalues of $\boldsymbol{X}^T\boldsymbol{X}$ and $V = \left[v_1,\dots,v_p\right]$ the corresponding set of eigenvectors.
3. The j-th eigenvector v_j is also the j-th PCA loading vector, therefore the j-th PC can be calculated as $\boldsymbol{X}v_j$.

The remaining steps to perform regression is straightforward. Let V_k consist of the first k columns of V, and let $T_k = \left[\boldsymbol{X}v_1,\dots,\boldsymbol{X}v_k\right]$ be the score matrix for the first k PCs. Then by using OLS to regress $\boldsymbol{y}$ on T_k, we can use equation 4.9 to estimate the regression coefficients β_k as $\hat{\beta}_k = \left(T_k^T T_k\right)^{-1} T_k^T \boldsymbol{y}$. Pre-multiplying by V_k then gives the regression coefficients for the original variables: $\hat{\beta} = V_k\hat{\beta}_k$.

Partial Least Squares

As mentioned already, PLS tries to find the derived variables that not only have maximum variance within $\boldsymbol{X}$, but also maximize their correlation with the output variable $\boldsymbol{y}$. The corresponding optimization problem for the m-th PLS direction $\hat{\varphi}_m$ is:

$$\begin{aligned} \max_{\alpha} \quad & \left(\mathrm{Cor}\left(\boldsymbol{y},\boldsymbol{X}\alpha\right)\right)^2 \mathrm{Var}(\boldsymbol{X}\alpha) \\ \text{subject to} \quad & \|\alpha\| = 1, \\ & (\boldsymbol{X}\alpha)^T \boldsymbol{X}\hat{\varphi}_l = 0, l = 1,\dots,m-1. \end{aligned} \tag{4.53}$$

Assume again that $\boldsymbol{X}$ and $\boldsymbol{y}$ have been standardized, and initialize $\boldsymbol{X}_0 = \boldsymbol{X}, \boldsymbol{y}_0 = \boldsymbol{y}$.

There are several popular variants that can compute the PLS directions, but the following procedure is essentially repeated for $m = 1,\dots,M-1$:

1. Compute the loading weights by $w_m = \boldsymbol{X}_{m-1}^T \boldsymbol{y}_{m-1}$, and then normalize it by $w_m \leftarrow \frac{w_m}{\|w_m\|}$.
2. Compute the score vector by $z_m = \boldsymbol{X}_{m-1} w_m$

3. Compute the X-loadings p_m, which are the regression coefficients from regressing $\boldsymbol{X}_{m-1}$ on z_m: $p_m = \boldsymbol{X}_{m-1}^T \frac{z_m}{z_m^T z_m}$.
   Similarly, the y-loading q_m is computed from regressing $\boldsymbol{y}_{m-1}$ on z_m: $q_m = \boldsymbol{y}_{m-1}^T \frac{z_m}{z_m^T z_m}$.

4. Orthogonalize (or 'deflate') $\boldsymbol{X}_{m-1}$ by subtracting the contribution of z_m: $\boldsymbol{X}_m = \boldsymbol{X}_{m-1} - z_m p_m^T$.
   Optionally, $\boldsymbol{y}_{m-1}$ can also be deflated: $\boldsymbol{y}_m = \boldsymbol{y}_{m-1} - z_m q_m$

So initially, the first PLS direction will have its largest contributions from those variables that are strongly correlated with $\boldsymbol{y}$. After that, the variables are continuously modified by subtracting the contributions from the previous score vectors, which means that they will not play a role when determining the next direction. For the following, let $\boldsymbol{W} = [w_1, \ldots, w_M]$, $\boldsymbol{T} = [z_1, \ldots, z_M]$, $\boldsymbol{P} = [p_1, \ldots, p_M]$ and $\boldsymbol{q} = [q_1, \ldots, q_M]$ be the matrix counterparts.
To interpret the regression coefficients in terms of the original variables, there is an additional hurdle compared with PCR. As consequent weight/loading vectors are computed from deflated matrices, the contributions from the original variables get masked. The first step to convert the PLS coefficients $\hat{\beta}_{PLS}^M$ (from regressing $\boldsymbol{y}$ on T) to the original coefficients $\hat{\beta}$, is to note that the score vectors T can be written in terms of the original variables with a linear equation (since they are just linear combinations):

$$T = \boldsymbol{X}_0 R, \tag{4.54}$$

where $R = \boldsymbol{W}\left(\boldsymbol{P}^T\boldsymbol{W}\right)^{-1}$ (see [20]).
Additionally, note that $\hat{\beta}_{PLS}^M = \boldsymbol{q}$. This is because if the input variables (i.e. z_i) are orthogonal, the coefficients from a multivariate regression are identical to those from separate univariate regressions.
When we regress $\boldsymbol{y}$ on T, we can then write:

$$\begin{aligned} & \boldsymbol{y} = T\boldsymbol{q} \\ \Leftrightarrow\ & \boldsymbol{y} = \boldsymbol{X}_0 R\boldsymbol{q} \\ \Leftrightarrow\ & \boldsymbol{y} = \boldsymbol{X}_0 \boldsymbol{W}\left(\boldsymbol{P}^T\boldsymbol{W}\right)^{-1}\boldsymbol{q}. \end{aligned} \tag{4.55}$$

Therefore, PLS approximates $\hat{\beta}$ by $\boldsymbol{W}\left(\boldsymbol{P}^T\boldsymbol{W}\right)^{-1}\boldsymbol{q}$.

Results

We had to make some slight adjustments to the standard procedure, as (repeated) cross-validation by itself was not enough to obtain good estimates for the hyper-parameter (amount of components used). The issue was that it generally over-estimated the amount of components needed, which means that over-fitting is still a problem. Therefore, in addition to simply applying CV to the maximum possible amount of components, we considered two heuristic procedures that decrease this upper limit:

1. First select the amount of components needed to explain 90% of the variance in $\boldsymbol{X}$, and then apply CV with this upper limit. The cumulative proportion of variance for the first k components in PCR is the proportion of their corresponding ordered eigenvalues: $\frac{\sum_{i=1}^k \lambda_i}{\sum_{i=1}^p \lambda_i}$, while in PLS we compute: $1 - \frac{\mathrm{Var}(\boldsymbol{X}_{k-1} - z_k p_k^T)}{\mathrm{Var}(\boldsymbol{X})}$.

2. The second procedure works the same, but we first select the amount that explains 90% of the variance in $\boldsymbol{y}$.
   If we again have k components, then in PCR their proportion of $\boldsymbol{y}$'s variance is: $1 - \frac{\mathrm{Var}(\boldsymbol{y} - \hat{\boldsymbol{y}})}{\mathrm{Var}(\boldsymbol{y})}$, where $\hat{\boldsymbol{y}}$ are the predictions from a model with k components.
   In PLS this can be computed as (assuming that $\boldsymbol{y}$ is also being deflated): $1 - \frac{\mathrm{Var}(\boldsymbol{y}_{k-1} - z_k q_k)}{\mathrm{Var}(\boldsymbol{y})}$.

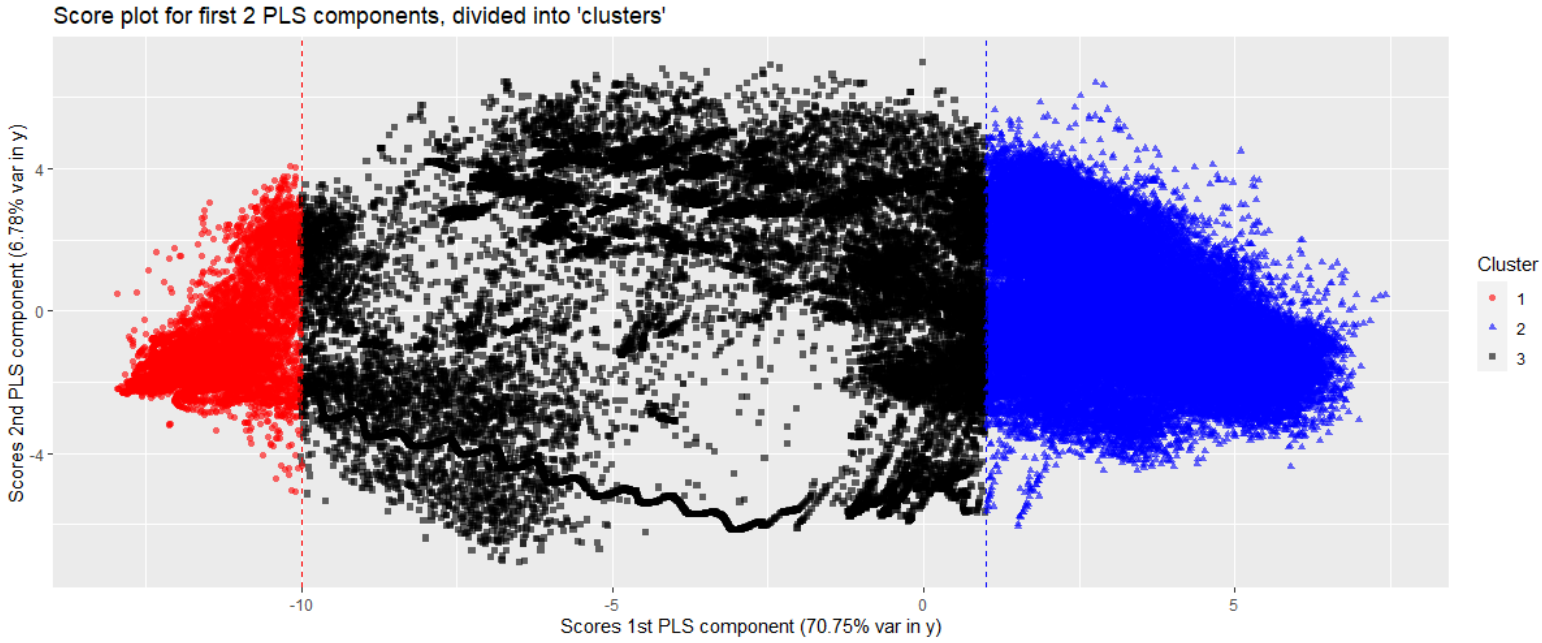
The results for the point predictions are shown in table 4.10, where we distinguish between the two models and the kind of heuristic method used. By first comparing the models, it is clear that PCR gives the best predictive results in general, but the differences between the two are mostly insignificant. This seems to support the earlier remark that while PLS may produce variables that are inherently more correlated with the output variable, it tends to have a higher variance. In this case, the decrease in bias that it brings is offset by a larger increase in variance.
Focusing on PCR, it is also clear that the heuristic procedures are not effective in improving predictive performance. The second procedure could not even be implemented, because the requirement of explaining 90% of the variance in $\boldsymbol{y}$ could not be satisfied. Furthermore, the first procedure returns nearly the same results. Intuitively, one might explain this lack of improvement by the fact that PCR constructs directions to explain variance in $\boldsymbol{X}$ first-hand, rather than in $\boldsymbol{y}$. Therefore, selecting the most important components would not necessarily increase predictive performance.

Predictive performance of dimension reduction models (without outliers)						
Method	Model					
	PCR			PLS		
	RMSE	MAE	R^2	RMSE	MAE	R^2
CV only	1.22	**0.86**	0.81	1.24	0.87	0.8
$\boldsymbol{X} \rightarrow CV$	**1.21**	0.88	**0.82**	1.24	0.87	0.8
$\boldsymbol{y} \rightarrow CV$	X			X		

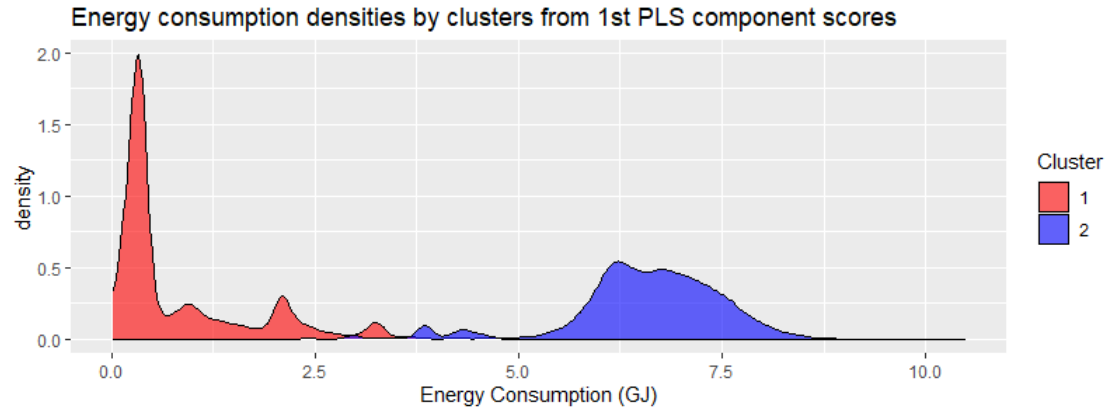
Table 4.10: Evaluation metrics for the two dimension reduction techniques, when outliers are removed. Under "Method" we employed repeated CV (10 repeats) with the previously described heuristic procedures to find the amount of components to use ("X" means that not enough components could be found to satisfy the 90% requirement).

Comparing the results with those of the OLS and regularized models, it is obvious that the current models are not capable of improving predictions in a meaningful way. Furthermore, we should avoid the use of heuristics, as they require knowledge of the test-data and are therefore meaningless in practical situations where we only have access to the training data. Of course, one could cross-validate the proportion of explained variance, but considering the poor results we have seen this is not something we will consider. Thus, the predictiveness of the dimension reduction methods will not be further explored.

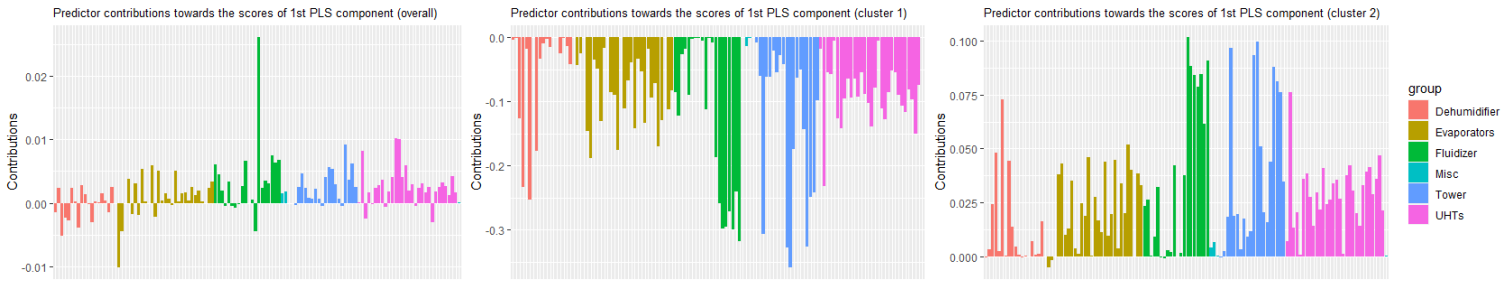
Predictive performance non-withstanding, the current models can still be used to gather some unique insights. Essentially, the current models decompose the outcome variable into a smaller set of derived variables, and therefore could provide more information about its dependencies. The first way to do this are score plots, as we show in figure 4.13. This is simply a scatter-plot of the score vectors formed from the first few PLS directions, or in other words it is a low-dimensional approximation of the original input space. We also show two crudely defined clusters, along with their separate energy consumption distributions and contributions from each variable in the first direction (i.e. the contribution for j-th variable is $\frac{\sum_{i=1}^{N} X_{ij} \cdot w_{1j}}{N}$, where w_{1j} is the j-th element of the first weight vector w_1). There is a clear distinction in energy profiles/variable contributions between the two clusters, with both clusters seemingly representing the active/non-active production states (as the energy consumption distributions suggest). This shows a possible application of the scores as a cluster analysis tool, which means that we could potentially use the input variables in this way to form clusters, rather than exclusively through the output variable as we did in section 3.4.
Aside from the scores, we can also visualize the loadings in *bi-plots*, as we show in figure 4.14. The loadings from the first two PLS components are shown for each variable as arrows, which represent both its importance (length) and its correlation with other inputs/output (similar/opposite/perpendicular directions mean positive/negative/no correlations). Therefore, these plots summarize the correlation structure and the importances of each variable, and can also be used to see if the inputs can be divided into homogeneous groups. For example, the variables for both evaporators and UHTs are well-separated from each other.
Finally, we show the variable importances in figure 4.15, which are defined by the weighted sum of the absolute regression coefficients, where the weights are a function of the decrease in error by the amount of components (see [45]). Compared with LASSO, the importance values seem to be more evenly distributed, both within and between the groups.



(a) Score plot with two crude clusters for illustration.



(b) Energy consumption distributions by cluster.

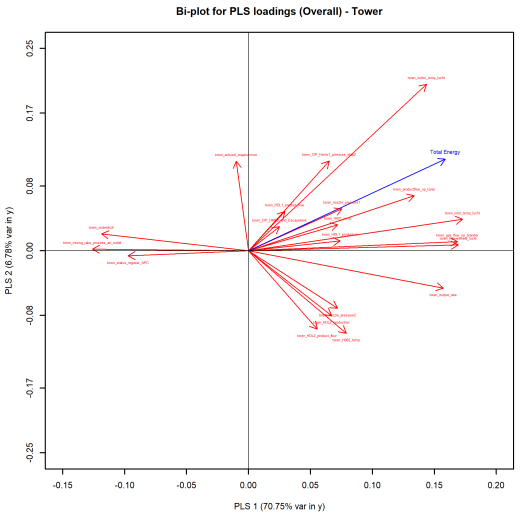


(c) Contributions (Overall)

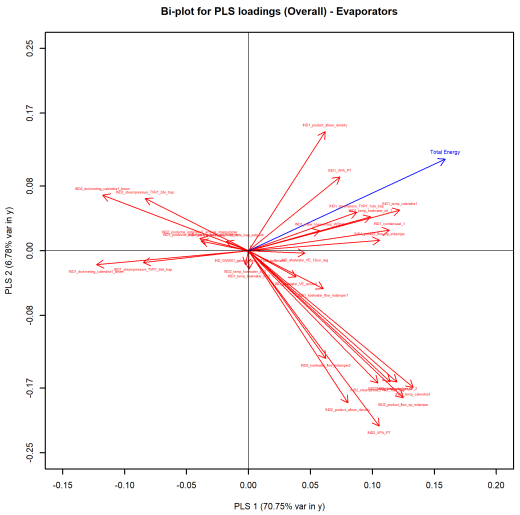
(d) Contributions (Cluster 1)

(e) Contributions (Cluster 2)

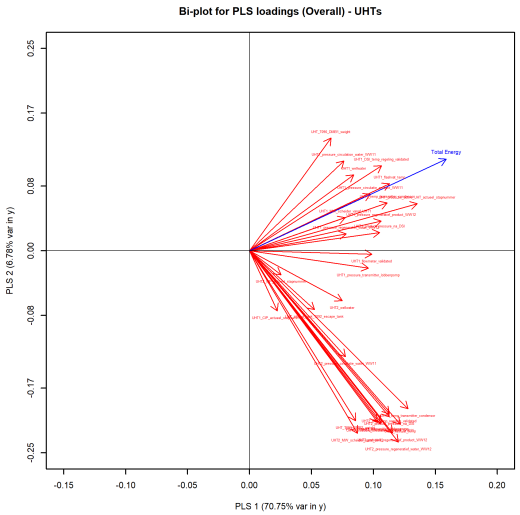
Figure 4.13: Score plot of the first two PLS components, along with separate visualizations for two crude clusters.



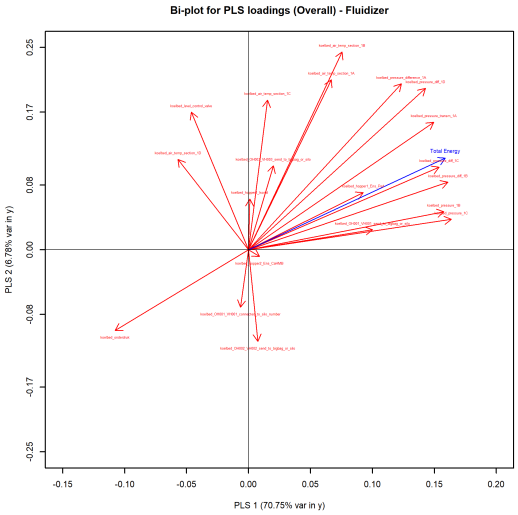
(a) Tower.



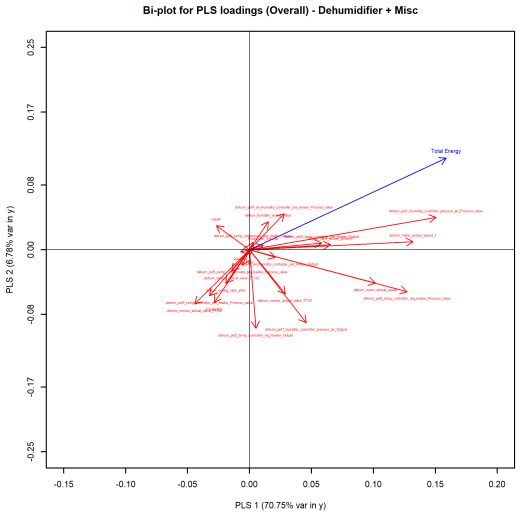
(b) Evaporators.



(c) UHTs.

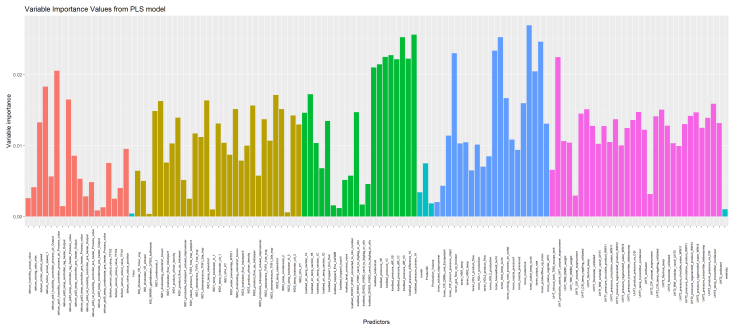


(d) Fluidizer.

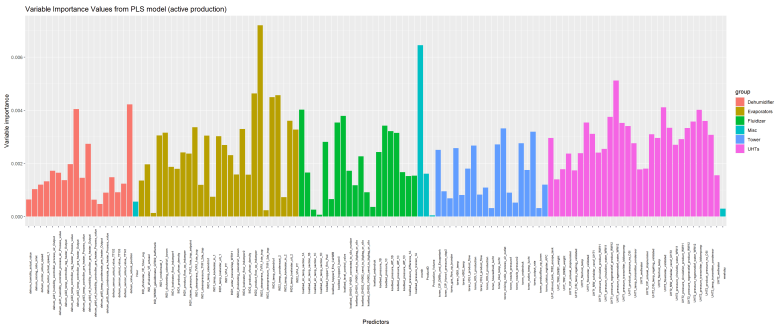


(e) Dehumidifier and miscellaneous.

Figure 4.14: Bi-plots for the first two PLS loadings directions. Input variables are separately shown for tower, evaporators, UHTs, fluidizer and dehumidifier by red arrows, while the output variable is denoted with a blue arrow.



(a) Overall.



(b) Active production.

Figure 4.15: Variable importance values from the PLS model for the overall situation and active production periods.

4.3. Discussion

In this chapter, we have considered several standard predictive models for the total energy consumption of the powder drying plant. Our primary goal was to find the most accurate model in terms of predictive performance over the test data, where for now we have limited our attention to several variants of the linear regression model. Furthermore, we explored how these models could be used to obtain more insight into the energy consumption, by studying whenever the energy efficiency changes (which was measured by the relative contribution of over-and underestimations from the model), and by visualizing the dependencies of the energy consumption.

To adequately assess the predictive performance, which we did for the overall data, periods of active production and transitional periods, we first had to study the effect of outliers. This was done for the reference model (taken to be OLS, i.e. linear regression with all variables included), and we logically concluded that we should exclude these outliers for better results.

Aside from OLS, we considered two additional classes of linear regression models. The first set were the penalized models (ridge, lasso and elastic net), which try to simplify the models by discarding unimportant variables. The second set were the dimension reduction models (PCR and PLS) which also try to reduce the complexity of the models, but they do this by combining variables rather than by shrinking their influence. The best results from each class of models for the overall data is summarized in table 4.11, where we highlighted the best results in bold. Starting with point accuracy, we measured the error through RMSE and MAE (average error in Giga-joules) and R^2 (squared correlation). We can see that on average the lasso model performs best. It would also appear that ridge regression returns very bad results, which we explained by its *joint shrinkage property*. Essentially, it tends to filter out correlated input variables together, regardless of importance. For our data-set, which contains many correlated inputs as we showed, it therefore may discard too much information contained in these variables. Based on these initial results, we henceforth focused exclusively on the lasso model.
The results for OLS and lasso during every kind of period, is summarized in tables 4.12a and 4.12b. In general, the predictions are most accurate during active production and long transition periods, due to them representing long and stable periods that see relatively little change. Vice-versa, the predictions are worst during short transitions, since it is more difficult to model large changes over a short period.
Comparing the models, we see that lasso provides a near universal improvement in point accuracy, even though it is slight. The results from the prediction intervals (PIs), constructed with an expected coverage rate of 95%, show a more two-sided story however. The best method for each model was first picked from a set of computational procedures, and eventually we decided to use quantile regression for OLS and the *split conformal-inference* method for lasso. These decisions were based on the following evaluation metrics: average width, coverage and the score function (which combines the contributions from width and deviations outside the PI). We see that while one approach provides a much smaller width and score, the other approach provides on average a nominal coverage, which is arguably the most important metric.

Predictive point accuracy (Overall and without outliers)			
Model	Metrics		
	RMSE	MAE	R^2
OLS	1.22	0.85	0.81
Ridge	5E52	2E50	1E-5
Lasso	**1.17**	**0.81**	**0.82**
PCR	1.22	0.86	0.81

Table 4.11: Point accuracy evaluation metrics for all the currently considered models, for the overall situation.

For the products, during periods of active production, one of the things we considered was using either the full training data or limiting it to the periods when the product was registered in the system. The idea was that by limiting the training data to intuitively more relevant data, the model could better pick up the specific patterns inherent to that product. While for some products this approach ended up being effective, it can also give very large errors for other products. Thus, on account of it being so unreliable, we recommend not limiting the training data to the specific product we want to make predictions for.
An alternative strategy which we employed to limit the training data to relevant parts, was limiting the training data to active production periods as well. In terms of point accuracy, this gave universally better results. However, for PIs the improvement was not so clear-cut, since we saw a decrease in all metrics alike. Consequentially, the coverage decreased below the nominal level of 95%. Therefore, to make simple point predictions we should limit the training data as described, whereas for PIs we instead recommend using the full training data. Both these configurations are used in the tables .

Predictive Point Accuracy							
Model	Metrics	Period					
		Overall	Act. Prod	Transition			
				Long	Medium	Short (diff)	Short (same)
OLS	RMSE	1.22	0.74	0.62	1.08	**2.23**	**1.85**
	MAE	0.85	0.57	0.37	0.75	1.5	**1.44**
	R^2	0.81	0.36	0.37	0.6	0.02	0.18
Lasso	RMSE	**1.17**	**0.69**	**0.61**	**1.07**	2.24	1.92
	MAE	**0.81**	**0.52**	**0.37**	**0.74**	**1.48**	1.49
	R^2	**0.82**	**0.38**	**0.39**	**0.6**	**0.04**	**0.2**

(a) Point evaluation metrics for the considered models, where under "Act. Prod" we limited both the training and test data to active production periods (as this gives the best results). Furthermore, the best results have been high-lighted in bold.

Predictive Interval Accuracy							
Model	Metrics	Period					
		Overall	Act. Prod	Transition			
				Long	Medium	Short (diff)	Short (same)
OLS	Width	**2.9**	**2.65**	**2.35**	**3.18**	**4.5**	**4.29**
	Cvrg	85.4	91	62.5	87.1	**92.9**	**95.2**
	Score	**4.41**	**3.7**	**3.05**	**4.16**	**8.63**	**5.68**
Lasso	Width	5.57	5.61	3.71	5.59	9.57	9.24
	Cvrg	**94.9**	**96.7**	**94.6**	**91.6**	90.5	93.4
	Score	7.18	6.28	3.83	7.03	18	11.4

(b) Interval evaluation metrics for the considered models, where under "Act. Prod" we only limited the test data to active production periods (as this gives the best results). Furthermore, the best results have been high-lighted in bold.

Table 4.12

Comparing the models during transitional periods, we see that in contrast to the other periods, the lasso model does not necessarily improve upon the OLS model (especially during short transition periods). For the PIs, we attributed this to the data-splitting step of split conformal inference, as smaller parts of the data (such as the short transition periods) are more susceptible to these random elements of the algorithm. Even with Multi S-CI, which is supposed to mitigate this problem, we only saw a slight improvement for product switches.

Another aspect that we assessed the performance by, was the model's generalisability to products it had not encountered before (which included *B953* as part of the top 10). In general, the accuracy for these products seemed to be worse compared with other products, indicating that the current models can not adapt well to new products. This provides another reason to consider more advanced models, which generally have higher adaptability.

To better understand how the prediction errors were composed, we made several visualizations. These made it clear that certain periods had unproportionally larger errors. In particular, September was clearly a consistent problem area, as we saw large prediction errors during this month for every scenario. Having concluded that the lasso model generally provides the best predictions among the treated models, we applied this model in a first attempt to answer the main research questions.

For the first research question, we had to assess whenever the largest changes in energy efficiency took place, which we did by calculating the relative contributions from over- and underestimations. Our justification for this is that the predictive model is trained on older data, and therefore represents an older period that we want to compare against. The predictions from the model therefore form a baseline, and if for example the recent, actual values are structurally smaller than this baseline, then this is an indicator that the efficiency has improved. An assumption is that the model can adapt sufficiently well to the data, as otherwise the results may be misleading (since the over- and underestimations could then be caused by the model's poor performance, rather than any structural changes in efficiency). To enforce this, we also calculated the contributions for periods of active production, since the predictive models have lower errors for these periods.
As for the contributions themselves, we had to calculate them in separate ways for prediction points and intervals. For the former, we simply calculated the MAE conditioned on predictions that were larger/smaller than the actuals. For the latter, we simply took the part of the score function attributable to over- and underestimations. Note that this only considers more extreme deviations (since they need to occur below/above the PI, rather than just a singular value), and therefore does not necessarily represent the full picture. In general, we had found that during active production, there was a much bigger contribution from underestimations, indicating that the efficiency may have decreased during these periods. On the other hand, the overall data showed a better balance between over-and underestimations, so any improvements in efficiency would have likely occurred during transition periods.
Further research should first repeat the analysis on more accurate models, not only to see if the results were due to the quirks of the model, but also to mitigate the impact of the model's own inaccuracies on the results. Furthermore we should do a more detailed analysis, by better understanding whenever the changes in efficiency occur and by comparing the contributions with those of the training data.

The second research question concerned itself with the dependencies of the energy consumption, which we interpreted as the variable importances (and their correlations). First, we visualized the overall variable importances for the lasso model by the 5 main parts of the drying process: evaporators, UHTs, tower, fluidizer and dehumidifier. We found that the most important variables came from the dehumidifier and tower, while for the other groups the importances

were more homogeneous. Similar conclusions apply to active production, but the importance of the tower-related variables increased.
Secondly, we visualized the dependencies through the PLS model (see figure 4.14), using so-called *bi-plots*. Essentially, each variable was represented by an arrow, whose direction denoted its correlation with other variables and its length the importance. This allows one to quickly visualize the correlation structure and importances of the variables, and an interesting conclusion was that there were some well-separated groups of correlated variables. This means that the energy consumption could potentially be represented by a smaller group of latent factors, or rather that the dependencies are comprised of a smaller group of variables.
Further research in this area should be to compute the variable importances for other models, and see whether common patterns occur.

A final remark is that initially, we also considered a separate linear model alongside OLS that included second-order interactions between the 5 most important variables. This was a rudimentary approach to include some more complicated patterns inside the model, which seemed to be effective in some aspects as certain evaluation metrics were slightly improved. This is another prior indicator that we should consider more advanced, non-linear machine learning models to improve the predictions, as these can capture more complicated patterns in the data. Therefore, we shall discuss these models in the next section.

5

Advanced Model Fitting

In the previous chapter, we started our predictive modelling with models that were essentially linear in nature. Namely, we considered the regular OLS model, its regularized variants and the dimension reduction methods, with the latter two aimed at reducing the (effective) degrees of freedom of the feature space to mitigate over-fitting. While the regularized models slightly improved predictive performance, it is clear that there is room for improvement since we showed that the data contained more complex patterns, which can not be captured by a standard linear model. Furthermore, it became clear that the data could be represented by a certain amount of clusters which provides another possible avenue for improvement, since each cluster might contain specific patterns that could be better captured by a model specifically fitted to it.

In this chapter, we will first consider some more advanced non-linear methods, which were chosen based on a literature review for similar prediction problems and the fact that they represent either a hybrid between linear and non-linear modelling, or are a full data-based non-linear approach. In particular the chosen hybrid model was MARS (Multi-variate Adaptive Regression Splines) model, and the data-based approach was Random Forest (RF) [36]. Other models which were also tested, were support vector regression, neural networks and gradient boosting machines, but the former was too computationally expensive and the latter two did not improve predictiveness (for the neural network this was presumably due to a lack of data).
The MARS model was chosen based on the fact that it represents a middle ground between linear models and data-based approaches, since it manages to retain the linear regression's simple additive form while still being able to incorporate higher order terms. Meanwhile, RF is a pure data-driven model, that is designed inherently to improve predictive performance. Aside from this, we also discuss our cluster-based approach, which for reasons already mentioned could potentially increase predictive performance by considering each cluster of the data separately.

5.1. Non-linear models

5.1.1. Theory

MARS

MARS is an adaptive regression procedure, which can be seen as a generalized version of forward stepwise regression. The underlying idea is that any regression surface can be represented by a sum of piece-wise linear basis functions, which are either of the form $(x-t)_+ = \max\{0, x-t\}$ or $(t-x)_+ = \max\{0, t-x\}$, where t is the knot that defines the piece-wise boundary. More specifically, a collection of *reflected pairs* is first constructed, by considering every variable and each of its unique values:

$$\mathbb{C} = \left\{ \left(X_j - t\right)_+, \left(t - X_j\right)_+ \right\}_{\substack{t \in \{x_{1j}, \ldots x_{Nj}\} \\ j=1,\ldots,p}}. \tag{5.1}$$

If $\mathbb{M}$ is the current set of functions used by MARS, which starts with only the constant function $h_0(x) = 1$, MARS proceeds in a stepwise forward fashion. It adds to $\mathbb{M}$ the product between an element $h_l(x)$ in $\mathbb{M}$ and a reflected pair in $\mathbb{C}$, i.e. $h_{l+1}(x) = h_l(x) \cdot (x_j - t)_+ + h_l(x) \cdot (t - x_j)_+$, such that it has the largest decrease on training error, whereby MARS takes the usual linear form:

$$f(X) = \beta_0 + \sum_{m=1}^{l+1} \beta_m h_m(X). \tag{5.2}$$

This process continues until $\mathbb{M}$ reaches some threshold M. Since the model will contain many terms, and therefore will be likely to over-fit, a backwards pruning-procedure is then applied to remove those terms whose removal causes the lowest

increase in residual squared error. The final amount is determined by minimization of the generalized cross-validation (GCV) criterium:

$$GCV(\lambda) = \frac{\sum_{i=1}^{N}(y_i - \hat{f}_\lambda(x_i))^2}{(1 - \frac{M(\lambda)}{N})^2}, \tag{5.3}$$

where $\hat{f}_\lambda$ denotes the model with size λ (number of terms) and $M(\lambda)$ measures the effective degrees of freedom [36].
A large benefit of this regression method is that it can automatically find the appropriate amount of terms (i.e. it has feature selection like lasso), including higher-order ones, and represent them in a convenient linear form such as in 5.2. This allows one to better visualize the dependencies for each variable, including their interactions.
To estimate variable importance, the change in the GCV estimate 5.3 is monitored for each predictor during the pruning process, and those variables which cause the largest accumulated decrease in GCV are considered important.

Random Forest

Before we can explain what a random forest does, we first need to explain what a decision tree is. A single decision tree T is a hierarchical structure comprised of binary splits, where at every split the splitting point s among a set of variables x_j is chosen such that it minimizes the sum of squared errors:

$$\min_{j,s}\left[\min_{c_1}\sum_{x_i\in R_1(j,s)}(y_i - c_1)^2 + \min_{c_2}\sum_{x_i\in R_2(j,s)}(y_i - c_2)^2\right]. \tag{5.4}$$

R_1 and R_2 denote the obtained regions, i.e. $R_1(j,s) = \left\{X \mid X_j \leq s\right\}$ and $R_2(j,s) = \left\{X \mid X_j > s\right\}$. In every region R_j, the tree takes a value c_j, which is the average of the output values in that region $\hat{c}_j = \frac{1}{N_j}\sum_{x_i\in R_j} y_i$ (as this minimizes the inner terms in 5.4).
The process is then repeated on each obtained region, until we get a final tree with regions/nodes that are called *terminal*. In this way the decision tree can naturally capture interactions between the variables. Also, as with the MARS model, care should be taken that the tree does not grow too large and starts overfitting. Therefore, a complexity criterion has to be minimized [36], which is defined as:

$$C_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha \mid T \mid, \tag{5.5}$$

where $N_m = \#\left(\{x_i \in R_m\}\right)$, $Q_m(T) = \frac{1}{N_m}\sum_{x_i\in R_m}(y_i - \hat{c}_m)^2$ and $\mid T \mid$ is the amount of terminal nodes in T. Therefore, α controls the complexity, and can be chosen through cross-validation. Regardless of this tactic however, single decision trees are still known to overfit the data (i.e. large variance), and random forests try to mitigate this issue by lowering the variance.
A random forest is a collection of decision trees $\{T_b\}_{b=1}^{B}$, where each tree is grown on a bootstrapped sample $\boldsymbol{Z}^*$ of the training data. This process is mostly the same as described before, aside from an important change: at each split not all the variables are considered, and instead only a subset of m variables is optimized over. After all the trees are grown, the average is taken as the new prediction: $\hat{f}_{rf}^{B}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$.
This procedure exploits the fact that by averaging B i.i.d. random variables, each with variance σ^2, their average has a smaller variance: $\frac{1}{B}\sigma^2$. Of course, the constructed trees are not necessarily independent, which is why their decorrelation is improved by selecting only m variables at each split.
To calculate the importance for a variable, the decrease on the sum of squared errors is recorded every time the variable is selected as the splitting variable. Importance is linked to the average decrease.

5.1.2. Results

Before discussing the results, we first describe the tuning strategies employed to fit the new models. Unlike with the linear regression models, non-linear models tend to have more hyper-parameters that need to be optimized for. Therefore we first evaluated their performance over a hyper-grid, which consists of all combinations between sets of pre-selected values, and then selected the configuration that provided the lowest cross-validated RMSE. The initial hyper-grids and final parameter values are shown in table 5.1.
For MARS, *degree* is the maximum degree of interaction and *nprune* is the maximum amount of terms after pruning. For RF, *num.trees* is the amount of decision trees, *mtry* the amount of considered variables at each split, *min.node.size* the minimum size of the terminal nodes, *replace* an indicator if the bootstrap for each tree should occur with replacement and *sample.fraction* is the proportion used for the bootstraps. Also note that for computational reasons, we did not consider the amount of trees as part of the hyper-grid, and instead chose it based on the OOB-curve ("out-of-bag"-curve) as shown in figure 5.1. The OOB-error for an observation is its average prediction error from those trees that were not trained on it, and therefore provides an efficient way to estimate the test error. In this case, we can use it to select an appropriate amount of trees before starting with the hyper-grid optimization, by choosing the amount of trees at which the error starts to stabilize. In this case, we found that 300 trees was a good amount.

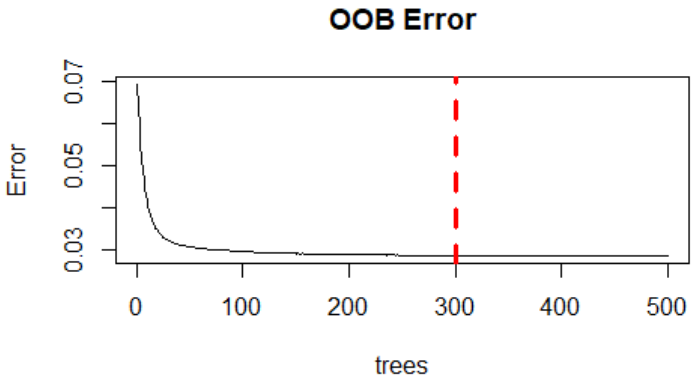


Figure 5.1: OOB error curve for random forest, with the vertical dashed line indicating when the error has stabilized.

Model	Parameter	Hyper-grid	Overall	Active Production
MARS	nprune	2, 12, 23, 34, 45, 56, 67, 78, 89, 100	56	89
	degree	1, 2, 3	3	3
RF	num.trees	X	300	300
	mtry	6, 18, 30, 39, 48, 122	48	39
	min.node.size	1, 3, 5, 10	1	1
	replace	TRUE, FALSE	FALSE	FALSE
	sample.fraction	0.5, 0.63, 0.8	0.8	0.8

Table 5.1: Considered hyper-grids for the non-linear models, along with the final chosen values for the overall and active production scenario's.

Now that we have found the appropriate parameters for the models, we first show the predictions for the overall situation in tables 5.2a and 5.2b. For comparison purposes, we also show the results for the lasso model. Furthermore, aside from adopting the previously discussed split-conformal inference method (S-CI) for producing 95% PIs, we also used a technique specifically tailored to RF called *Quantile Regression Forests* (QRF) [54]. The idea is essentially the same as for quantile regression, as in that we try to estimate the conditional quantiles rather than the conditional mean. However, instead of optimizing the quantile loss function, it does this by making a very straightforward modification to the usual RF prediction procedure. Instead of estimating the conditional mean like RF, QRF first estimates the conditional CDF $F(y \mid X = x)$ and then uses this estimation to estimate the conditional α-th quantile $Q_\alpha(x)$, as summarized in the following steps:

a) Grow k trees $T(\theta_t)$, $t = 1,\dots,k$ as in RF. However, all observations in every leaf l from every tree should be saved.

b) For a given $X = x$, let $R_{l(x,\theta)}$ be the rectangular subspace taken by leaf l from a single tree $T(\theta)$. Then for every tree $T(\theta_t)$ and observation $i \in \{1,\dots,N\}$, compute the weight $w_i(x,\theta_t) = \frac{\mathbb{1}_{\{X_i \in R_{l(x,\theta)}\}}}{\#\{j : X_j \in R_{l(x,\theta)}\}}$.

c) Average the previous weights over all trees: $w_i(x) = \frac{1}{k}\sum_{t=1}^{k} w_i(x,\theta_t)$.

d) Estimate the conditional distribution as $\hat{F}(y \mid X = x) = \sum_{i=1}^{n} w_i(x)\mathbb{1}_{\{Y_i \le y\}}$.

e) Finally, estimate the conditional quantile as $\hat{Q}_\alpha(x) = \inf\{y : \hat{F}(y \mid X = x) \ge \alpha\}$.

Even though QRF works in largely the same ways as RF, it is shown in [54] that this simple modification still provides asymptotically consistent estimates of the quantiles, since $\hat{F}(y \mid \boldsymbol{X} = x)$ itself is asymptotically consistent. This holds under the following non-stringent assumptions:

1. The proportion of values in a leaf, relative to all values, is vanishing as $n \to \infty$.
2. The minimal number of values in a tree node is growing as $n \to \infty$.
3. When finding a variable for a split-point, the probability that variable $m = 1,\dots,p$ is chosen for the split-point is bounded from below for every node by a positive constant, i.e. every variable is accounted for.
4. If a node is split, the split is chosen so that each of the resulting sub-nodes contains at least a proportion γ of the observations in the original node, for some $0 < \gamma \le 0.5$.
5. The true CDF of the predictive distribution is Lipschitz continuous.

Moving on to the results, it is clear from the point predictions that the non-linear models are capable of improving the predictions by quite a bit compared to lasso, meaning that the input-data indeed contains higher order terms that can be used to improve predictions. In summary, RF produces the most accurate point predictions, although the MARS model does not lag far behind.
When considering the interval results however, the non-linear models are struggling to provide PIs with nominal coverage. For the S-CI method, this can be explained directly from its algorithm as described in Algorithm 3. As a reminder, this algorithm estimates the predictive uncertainty by accumulating two quantities:

1. The predicted residual for a new observation $\hat{\rho}(\boldsymbol{x}_{test})$, where $\hat{\rho}$ is a separate regression model for the residuals from the regular regression model $\hat{\mu}$.
2. The kth smallest value from the set of validated, scaled residuals $\left\{\frac{|y_{train}(i) - \hat{\mu}(x_{train}(i))|}{\hat{\rho}_N(x_{train}(i))} : i \in A_2\right\}$, where A_2 was the validation set and $k = \lceil\left(\frac{N}{2} + 1\right)(1-\alpha)\rceil$.

The issue that occurs with the non-linear models is that they underestimate these quantities, since they are more accurate and therefore produce smaller residuals. Therefore, both of the mentioned quantities decrease too much and underestimate the predictive uncertainty, which in turn causes the PIs to be too small with a coverage that is far below nominal. And aside from that, we also see the same non-robustness issue that we had with OLS in MARS (when using regular S-CI): some residuals end up blowing up the average width and score, although this is prevented by using Multi S-CI. It seems that this is also prevented with the RF model, but of course the undercoverage is most serious for this model due to the fact that it has the best predictive accuracy as explained.

Predictive point accuracy (Overall and without outliers)			
Model	Metrics		
	RMSE	MAE	R^2
LASSO	1.17	0.81	0.82
MARS	0.95	0.61	0.88
RF	**0.89**	**0.49**	**0.89**

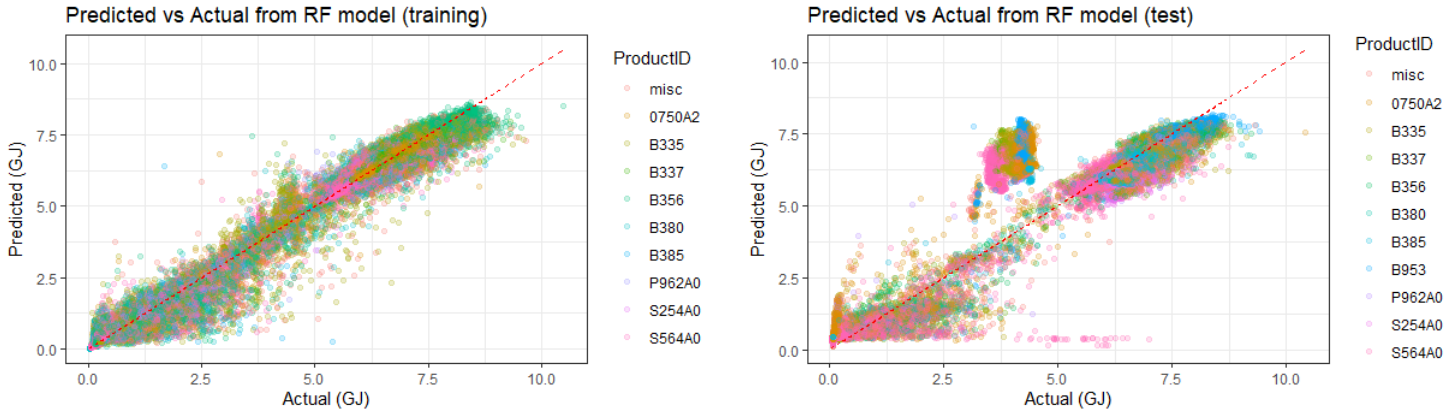
(a) Overall point accuracy.

Predictive interval accuracy (Overall)				
Model	Method	Metrics		
		Width	Coverage	Score
LASSO	S-CI	5.57	**94.9**	7.18
MARS	S-CI	5E37	81.4	5E37
	Multi S-CI	1.23	74	8.65
RF	S-CI	**1.2**	72.9	8.69
	Multi S-CI	1.36	76.1	8.52
	QRF	2.59	87.1	**6.83**

(b) Overall PI accuracy, where "S-CI" denotes split conformal inference and "QRF" is quantile regression forest. For S-CI, the results were averaged over 10 runs.

Table 5.2

For QRF the under-coverage is mitigated since it does not solely rely on residuals like S-CI does, and of course is a more robust method due to its focus on quantiles. It therefore produces better intervals, but still has too low coverage, which means that the conditional distributions it uses may not be applicable for new data. As we show when only considering active production, or when considering the cluster-based approach in the next section, we can improve this by fitting a separate QRF model on "easier" subsets of the data, therefore better guiding the models into what kind of conditional distributions should be used.
As an additional confirmation that the RF model better adapts to the data, we again show some diagnostic plots in figure 5.2. On the plots showing the predicted against actual values, we see for the training and test data alike that the points are closer to the diagonal red line as compared with the OLS model. However, for the RF model we still see some kind of misfit in the middle portion of the test data (just as we did with the OLS model), as there is a clear group of points situated above the diagonal. This group of observations likely represents the transitory periods, as these are more difficult to predict for. The time-plots of the residuals also show a clear improvement, since they are in general smaller and less often show a sudden change in residual variance (indicating that the heterogeneity aspect is improved). Furthermore, as with the OLS model, we see many negative residuals during September of 2021, i.e. there are many overestimations during this month. From the QQ-plots, we can furthermore see that there are still some tails that are heavier than expected, although this issue has clearly been mitigated for the test data.

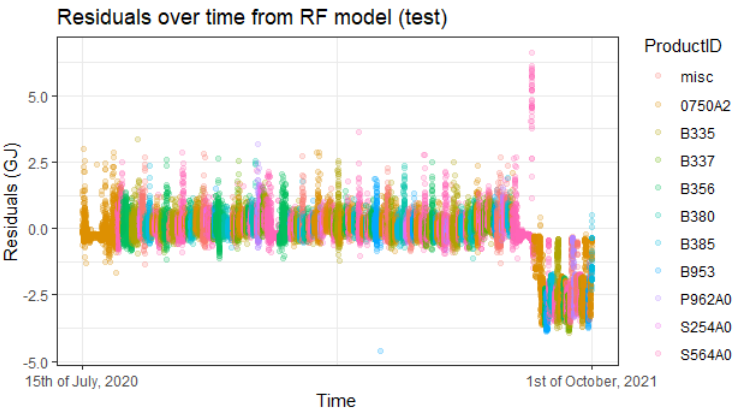


(a) Predicted values from the RF model plotted against actual values, for the training data.

(b) Predicted values from the RF model plotted against actual values, for the test data.

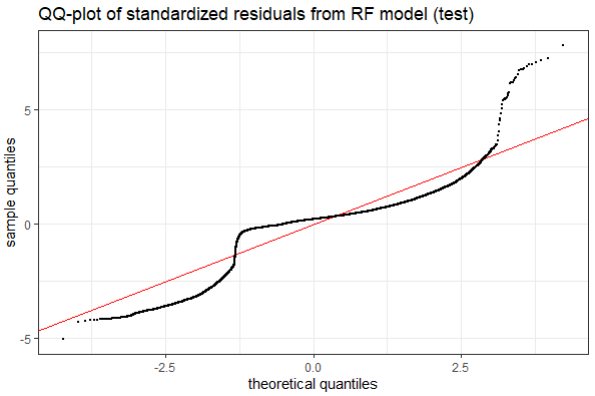
Figure 5.2

(c) Residuals over the training period for the RF model.



(d) Residuals over the test period for the RF model.

(e) Sample quantiles of the residuals from the training data are compared with the theoretical quantiles from a normal distribution, to assess normality.



(f) Sample quantiles of the residuals from the test data are compared with the theoretical quantiles from a normal distribution, to assess normality.

Figure 5.2: Diagnostic plots for the RF model. The colored points denote the top products.

The results for every period are summarized in tables 5.3a and 5.3b, and as before the results for the top 10 products are shown in Appendix D. Note that for specific products we only consider full training data sets, as limiting them to the specific products again turned out to be unreliable, similarly to the linear regression models.

In terms of point accuracy, the LASSO is clearly out-performed by the non-linear models, indicating the presence of non-linear patterns in the data. In particular the RF model generally performs best, with the largest improvements for the short and medium-term transition periods, therefore bringing the performance during such periods much closer to those of other periods. The long transition periods on the other hand only show a small improvement, and in terms of interval accuracy we can even see that all methods aside from the LASSO/S-CI combination struggle to provide a nominal coverage. This is in contrast to active production, as QRF then clearly produces the most accurate intervals. Compared with lasso, it produces intervals that are on average around 58.6% smaller (i.e. less than half the size), while still retaining the exact same coverage. A similar improvement is visible for the short transitions.

A common thread between these long and stable periods however, are the low score values. So ignoring the structural misfit that QRF seems to have for long transition periods, it could still be beneficial when its conditional distributions are less disturbed by noise or observations from distinct parts of the dataset, which is supported by the results from our cluster-based approach in the next section.

As for the top 10 products, it was also clear that there are some products for which RF still performs worse than MARS, for example RF has worse point accuracy for *S254A0* and *P962A0*. Therefore, RF does not bring an universal improvement, and other models might have to be considered to obtain the best results for a specific product. Also, the new products alongside *0750A2* still remain difficult to predict.

In conclusion, since it is clear that the RF provides the best point accuracy, and among the non-linear models it provides the best PIs with QRF, we will primarily focus on the RF model from now on.

Predictive Point Accuracy							
Model	Metrics	Period					
		Overall	Act. Prod	Transition			
				Long	Medium	Short (diff)	Short (same)
Lasso	RMSE	1.17	0.69	0.61	1.07	2.24	1.92
	MAE	0.81	0.52	0.37	0.74	1.48	1.49
	R^2	0.82	0.38	0.39	0.6	0.04	0.2
MARS	RMSE	0.95	0.57	0.59	0.87	1.17	1.35
	MAE	0.61	0.32	**0.34**	0.62	0.88	1.04
	R^2	0.88	0.43	0.46	0.73	0.24	0.27
RF	RMSE	**0.89**	**0.53**	**0.59**	**0.72**	**1.03**	**1.03**
	MAE	**0.49**	**0.28**	0.38	**0.54**	**0.6**	**0.67**
	R^2	**0.89**	**0.5**	**0.5**	**0.81**	**0.4**	**0.64**

(a) Point evaluation metrics for the non-linear models, where under "Active Production" we limited both the training and test data to active production periods (as this gives the best results). Furthermore, the best results have been high-lighted in bold.

Predictive Interval Accuracy								
Model	Method	Metrics	Period					
			Overall	Act. Prod	Transition			
					Long	Medium	Short (diff)	Short (same)
Lasso	S-CI	Width	5.57	5.61	3.71	5.59	9.57	9.24
		Cvrg	**94.9**	**96.7**	**94.6**	**91.6**	90.5	93.4
		Score	7.18	6.28	3.83	7.03	18	11.4
MARS	S-CI	Width	5E37	3.64	3E38	5.61	5.36	4.85
		Cvrg	81.4	93.8	49.4	88.1	89.7	81.3
		Score	5E37	4.57	3E38	6.6	8.29	10.3
	Multi S-CI	Width	1.23	1.02	1.23	2.33	2.13	2.13
		Cvrg	74	89.7	37.9	71.6	88.3	80.7
		Score	8.65	3.34	3.37	4.32	9.59	11.3
RF	S-CI	Width	**1.2**	**0.98**	**1.05**	**2.28**	**2.09**	**2.08**
		Cvrg	72.9	88.1	35.9	71.2	87.8	80.2
		Score	8.69	**3.32**	3.44	4.31	9.61	11.4
	Multi S-CI	Width	1.36	1.09	1.24	2.68	2.31	2.3
		Cvrg	76.1	90.7	42.1	76.2	88.7	81
		Score	8.52	3.36	3.18	4.35	9.45	11
	QRF	Width	2.59	2.32	2.74	3.49	3.59	3.84
		Cvrg	87.1	96.6	80.2	88	**92**	**95.8**
		Score	**6.83**	4.04	**3.02**	**3.8**	**7.02**	**5.07**

(b) Interval evaluation metrics for the non-linear models, where under "Act. Prod" we limited only the test data to active production periods (as this gives the best results). Furthermore, the best results have been high-lighted in bold.

Table 5.3

The predictive performance is visualized in figure 5.3. Comparing the results by month with those of lasso in figure 4.8, we still see the worst performance during August and September, although it would seem that the predictions during June have considerably improved, therefore eliminating this month as a problem area. For the overall data, it is also clear why QRF did not have a nominal coverage, since the performance during August/September and the second ordered consumption group is much worse than with lasso/S-CI.
The results for the products and ordered energy groups also seem to support some of the same conclusions as before, i.e. we have the worst performance for the new products, product *0750A2* and for the second ordered group.

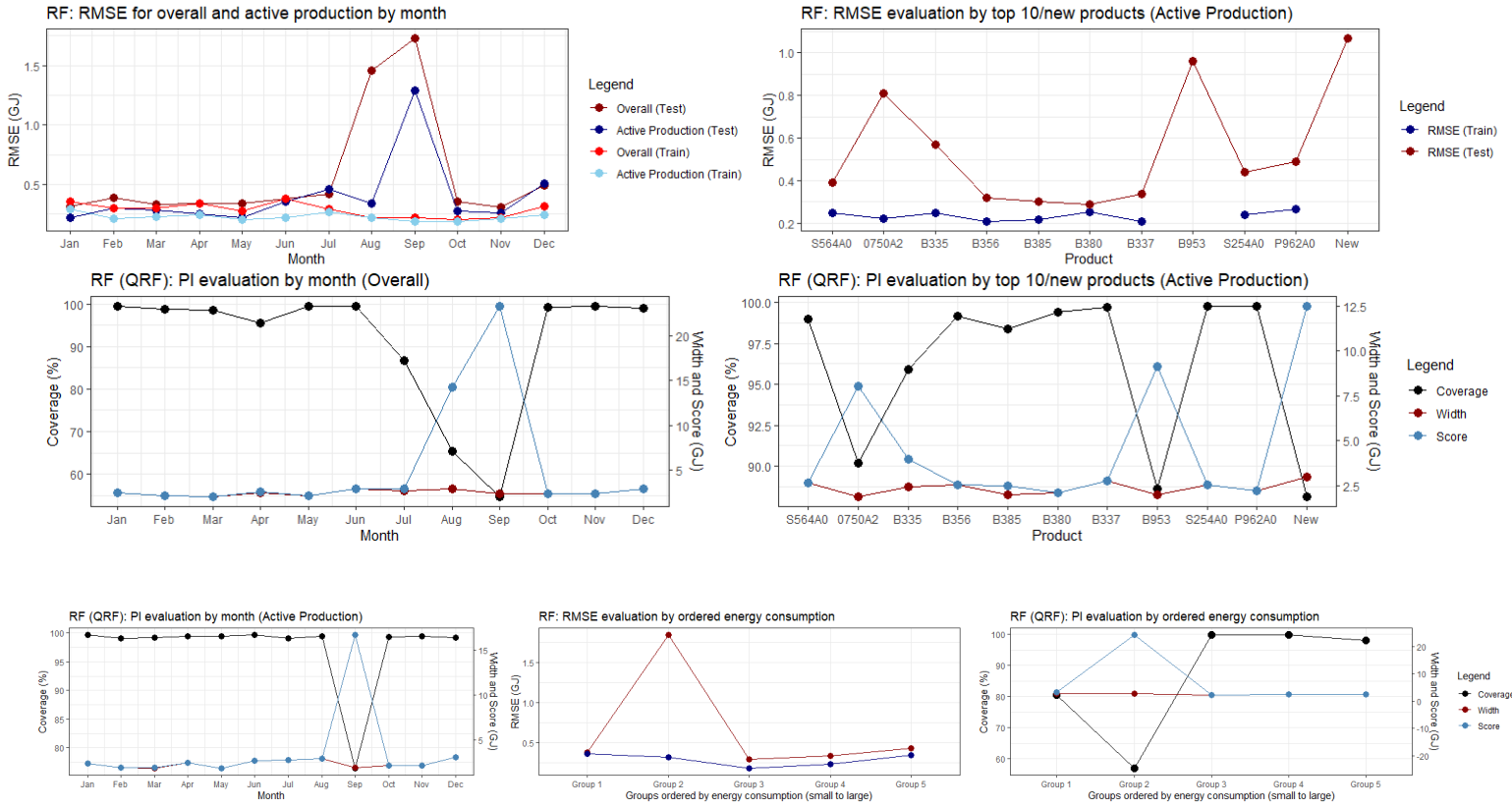
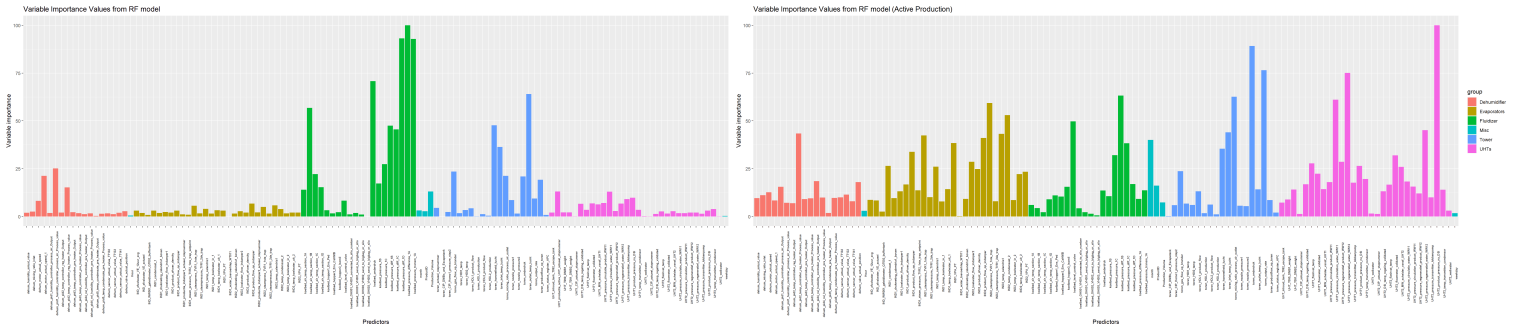


Figure 5.3: Predictive performance of the RF model, shown for the top 10 products, every month and ordered energy groups.

Finally, we quickly show the variable importances in 5.4, where we colored them by every distinct part of the drying process as before. Compared with lasso's variable importances, we see that the dehumidifier has decreased in importance, while the opposite applies for the fluidizer. As before however, the variables from the tower remain a relatively important group. Furthermore, it can be seen that the importances during active production have become more homogeneous, with the average importances of every group becoming closer to each other.



(a) Overall.

(b) Active production.

Figure 5.4: Variable importance values from the RF model for the overall situation and active production periods.

5.2. Cluster-based Approach

On prior occasions, namely when conducting data-exploration and when we considered the PLS model, we tried to make some preliminary assessments as to whether the data could be split into separate clusters. During data-exploration we considered daily energy consumption profiles, and applied some basic clustering algorithms to these profiles in order to find out whether the corresponding groups had characteristic patterns. Also, by comparing the energy consumption distributions, we found some evidence of structural differences between the groups.
A more clear picture was obtained when we first projected the complete data into a lower dimensional plane using PLS, and then considered parts of the data which were located in the left-and righthandside of the projected data. When we again compared the energy distributions, the difference was much clearer, therefore providing stronger evidence that the data could be clustered into different groups, with each group representing some distinct part of the production process.
Based on these prior promising results, we now formalize a new predictive approach, which makes use of some kind of clustering to improve predictions.
The idea is that by fitting separate models to each cluster, we can make sure that each such model can better capture the specific characteristics inherent to that cluster. This could be preferable to fitting a single model to the whole data, because a single model can be more easily "confused" by the range of different scenario's that the data represents. This is why we saw predictions improve when considering active production periods, since the energy consumption becomes more stable and easily predictable during such periods. The cluster-based approach is essentially an extension of this idea, where we now automate the finding of such "easy to model" subsets to a clustering algorithm.
The cluster-based approach consists of the following steps:

1. Either transform the data into a lower dimensional representation with PLS as explained earlier, or divide the data into shifts of 8 hours (such that each data-point is a sequence of energy consumption values over 8 hours).
2. Use a clustering algorithm to divide the training data into clusters.
3. Assign the test-data into the clusters, for example by assigning each point to the nearest cluster center.
4. If necessary, translate the cluster assignments back to the original data.

The motivation behind the second transformation in the first step comes primarily from the fact that we are dealing with production-related data, which naturally occurs in shifts of 8 hours (and therefore the transition periods between shifts are likely candidates for when the production state changes).
In terms of clustering algorithms, we considered the standard partitional and hierarchical algorithms that we used during data-exploration, and in addition we used a density-based technique called *DBSCAN* [26]. For a precise description on how DBSCAN works, see appendix B.2.3. It turns out that DBSCAN has a very desirable property, as in that it can separate noise from well-defined dense clusters, which will turn out to be very beneficial for the shifts-based clustering as we will discuss.
DBSCAN requires two parameters to be specified before-hand, which are ϵ (maximum radius of a cluster) and *MinPtns* (minimum amount of observations that a cluster must contain). Since we can not use domain knowledge to pick the right values, we will instead use heuristic procedures as described in [62]. To pick the appropriate values, we used a k-nearest neighbour plot for ϵ as shown in figure 5.5, where it is recommended to take $k = 2 * dimensions - 1$, while for *MinPnts* we simply chose $k+1$. The k-NN distance plot calculates the distances between every point and their k-nearest neighbour,

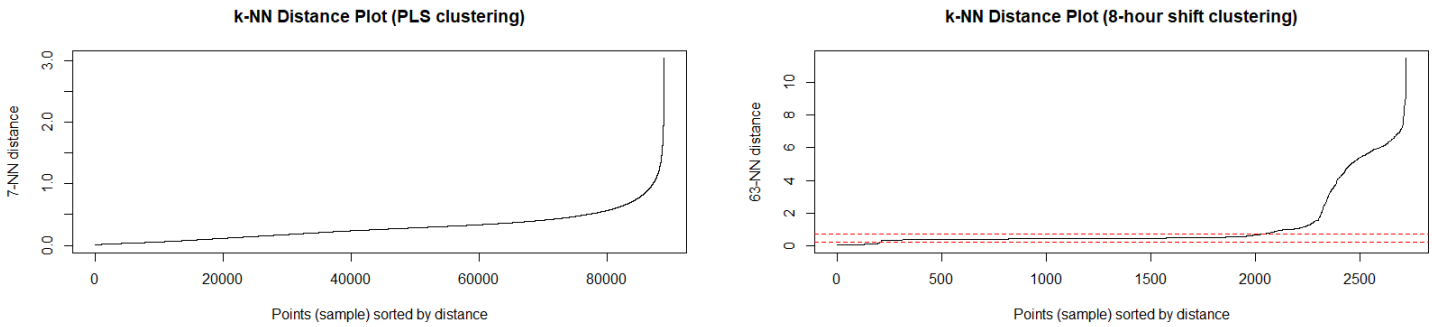


Figure 5.5: k-NN distance plots for the PLS and shifts-based clustering, which are used to find the threshold ϵ-value between noise and clusters. For shifts-based clustering, the final chosen elbow-point is highlighted between the horizontal dashed lines.

and then plots the sorted distances in ascending order. To find a good value for ϵ, we need to take the distance at its elbow point, i.e. the point at which the distance suddenly increases. The idea is that such a point provides a good threshold value between noise observations and well-defined clusters, since noise observations will usually have larger distances between each other than this threshold value, and will therefore not form part of a cluster. For both of the plots in figure 5.5 however, such an elbow point is difficult to find, since either the distances always increase (PLS) or there are multiple elbow points (shifts). Therefore, for PLS we did not consider DBSCAN, while for the latter we simply experimented with every elbow point. Eventually, we picked the value that visually gave the best-looking clusters for the training data, and used this as an upper limit for ϵ (which turned out to be 0.75).
In practice, in order to choose among a set of clustering configurations, we would have to assess them both formally as visually. Especially visualization is important, since it remains the best way to see if the clusters are compact and well-separated ("well-defined" clusters), and if the noise is separated from the actual clusters. For the formal assessment, we will use the *S_Dbw* CVI [49], which similarly to DBSCAN is density-based.
First, we explain our choice of this CVI, rather than the Dunn or Silhouette index that we defined in equations 3.5 and 3.6. As will become clear with the shift-based clustering, the clustered data will suffer from two issues that the aforementioned CVIs will struggle with, namely noise and "sub-clusters" (clusters which are closely located together). Beginning with the Dunn index, it is highly sensitive to both of these issues, since it simply uses the minimum pairwise distance between elements of clusters to measure the inter-cluster separation. In the presence of noise for example this can decrease sharply, which therefore causes the Dunn index to change dramatically and the corresponding optimal clustering result to be heavily dependant on the noise.

The silhouette index is a bit more robust to noise, since it uses the minimum *average* distance between clusters (defined by $b(i)$ in 3.5) to measure the separability, rather than simply taking the minimum. However, this is still susceptible to sub-clusters, since the inter-cluster separation will increase when sub-clusters are considered as one cluster. Therefore, the silhouette index can still give non-optimal results in such situations.
Our choice for *S_Dbw* is based on the empirical study conducted in [49], where it was shown that this CVI was the only metric that still gave good results, both in the presence of noise and sub-clusters. A quick description of *S_Dbw* is as follows. As with Dunn and Silhouette, *S_Dbw* uses notions of cluster compactness and separability. The compactness is defined by the intra-cluster mean dispersion $\mathbb{S}$:

$$\mathbb{S} = \frac{\frac{1}{K}\sum_{k=1}^{K}\|\mathbb{V}^k\|}{\|\mathbb{V}\|} \quad \text{(Cluster compactness)} \tag{5.6}$$

where K is the amount of clusters, $\mathbb{V} = \big(\text{Var}(X_1),\ldots,\text{Var}(X_p)\big)$ is the vector of variances for each variable, and $\mathbb{V}^{\mathbb{k}}$ is similarly defined for points in cluster C_k.
To measure separability, a limit value $\sigma = \frac{1}{K}\sqrt{\sum_{k=1}^{K}\|\mathbb{V}^k\|}$ is first calculated, and then used to define the relative density of a point x_i to two clusters C_k and $C_{k'}$, which measures how many points in their union are close to x_i:

$$\gamma_{kk'}(x_i) = \#\Big\{x_j : x_j \in C_k \cup C_{k'} \text{ and } d(x_i,x_j) < \sigma\Big\}. \tag{5.7}$$

The separability is then calculated as the average of the between-cluster densities:

$$\mathbb{G} = \frac{2}{K(K-1)}\sum_{k<k'} R_{kk'} \quad \text{(Cluster separability)} \tag{5.8}$$

where $R_{kk'} = \frac{\gamma_{kk'}(H_{kk'})}{\max\big(\gamma_{kk'}(\hat{C}_k),\gamma_{kk'}(\hat{C}_{k'})\big)}$ measures the quotient between the relative densities of the cluster centers $\hat{C}_k, \hat{C}_{k'}$

and their mid-way point $H_{kk'}$. Finally, the S_Dbw index is then simply the sum of $\mathbb{S}$ and $\mathbb{G}$:

$$S\_Dbw\text{-index} = \mathbb{S} + \mathbb{G}, \tag{5.9}$$

and should therefore be minimized. We summarize the results for different clustering configurations in table 5.4, and visualize the clusterings in figure 5.6 for the shift-based clustering. Note that we show the results for the first two PC components, because S_Dbw was undefined in many instances for the original data. Nevertheless, the results should still represent the full data well, since the PC components explain more than 90% of the variance.

Transform	Method	K / ϵ				
		2 / 0.25	3 / 0.375	4 / 0.5	5 / 0.625	6 / 0.75
PLS	k-Means	**0.41**	1.14	1.65	1.73	1.78
Shifts	k-Means	**0.41**	0.55	0.51	1.44	0.62
	Hier. (Complete)	**0.62**	2.63	0.8	2.26	3.35
	Hier. (Average)	0.54	**0.42**	1.65	2.87	4.41
	DBSCAN	1.31 (1)	3.88 (0.28)	1.92 (0.12)	1.85 (0.13)	**1.31**(**0.03**)

Table 5.4: Clustering evaluation results for the PLS and shifts-based clustering (first two PC components), using the S_Dbw internal measure. For DBSCAN, the values between parentheses indicate when noise has been removed. Furthermore, K denotes the amount of chosen clusters for k-means/hierarchical clustering, while ϵ denotes the cluster radius parameter for DBSCAN.

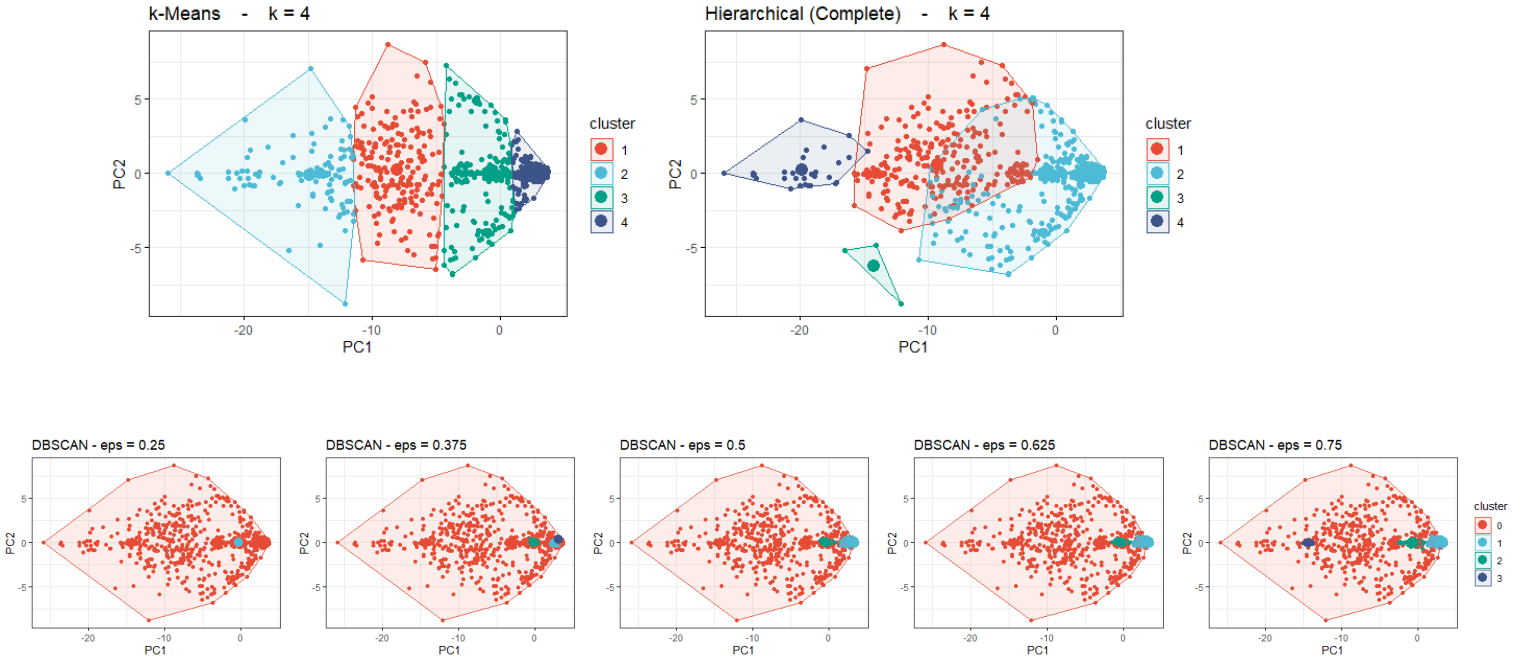


Figure 5.6: Several clustering methods applied to the PC-representation of the 8 hour shift data. For DBSCAN we show the influence of increasing ϵ.

In table 5.4, we omitted hierarchical clustering / DBSCAN for the PLS data, because of computational complexity and visually unappealing results. It also seems that the PLS data is more difficult to cluster on, so aside from the k-NN distance plots, this is another indication that the shift-transform allows better clusters to be formed. Furthermore, by considering both the tabulated results and figure, it is clear that hierarchical clustering keeps performing badly for the shift data, so we will not consider it from now on.

If we consider DBSCAN, it is also clear that by ignoring the noise, we can produce clusters that are much better defined compared with other algorithms. Looking at the visualization, it is clearly visible how well DBSCAN can separate noise from the proper clusters, in contrast to k-Means and hierarchical method. However, it also shows how sensitive it is to the chosen value of ϵ, which high-lights that this parameter should be carefully chosen. It thus seems that the shift-transform, together with DBSCAN, provides a good way to cluster the data.

To see whether this preliminary assessment holds up in practice, we show the predictive results using the same clustering configurations in tables 5.5 and 5.6 for the overall situation. Indeed, the shifts-based clustering tends to produce much

better results than PLS, and when looking at the point accuracy we see that the best results for RF are obtained with DBSCAN, while LASSO works best with k-Means. If we had selected the configuration using the *S_Dbw* metric together with the visualizations as described, we would have found the ideal results for RF but not for LASSO. This is likely because LASSO can not adopt well to the group of noise observations, whereas with k-Means the noise would have been spread over the groups, therefore mitigating their impact on the overall results. Furthermore, the *S_Dbw* metric does not work well with k-Means, giving a completely wrong indication of the amount of clusters to use. This is again likely caused by the noise perturbing the results too much. This is not a problem with DBSCAN of course, since we can simply disregard noise first and then evaluate, so it is easier to apply this metric with DBSCAN.

Point accuracy for Cluster-based Approach (Overall and without outliers)																	
K			2			3			4			5			6		
Metric			RMSE	MAE	R^2	RMSE	MAE	R^2	RMSE	MAE	R^2	RMSE	MAE	R^2	RMSE	MAE	R^2
Transform	Method	Model															
PLS	k-Means	Lasso	0.89	0.61	0.89	0.92	0.6	0.89	6E43	1E42	1E-3	6E56	1E55	1E-3	1.15	0.61	0.82
		RF	0.88	0.5	0.89	**0.88**	**0.49**	**0.9**	0.88	0.5	0.89	0.89	0.5	0.89	0.89	0.49	0.89
Shifts	k-Means	Lasso	0.97	0.64	0.87	0.84	0.37	0.9	0.7	0.34	0.93	0.66	**0.33**	0.94	0.69	0.33	0.93
		RF	0.86	0.47	0.9	0.65	0.38	0.94	0.65	0.38	0.94	0.63	0.37	0.95	**0.6**	0.36	**0.95**
	ϵ		0.25			0.375			0.5			0.625			0.75		
	DBSCAN	Lasso	1.24	0.77	0.81	0.76	0.41	0.92	0.81	0.4	0.91	0.78	0.38	0.92	0.93	0.39	0.88
		RF	0.72	0.4	0.93	0.45	0.28	0.97	0.42	0.25	0.98	0.42	0.25	0.98	**0.42**	**0.25**	**0.98**

Table 5.5: Point accuracy for the PLS and shifts based clusterings, evaluated with either k-means or DBSCAN for the overall data. Furthermore, K denotes the amount of chosen clusters for k-means, while ϵ denotes the cluster radius parameter for DBSCAN. The best results are highlighted for every clustering and method separately.

Interval accuracy for Cluster-based Approach (Overall)																	
K			2			3			4			5			6		
Metric			Cvrg	Width	Score	Cvrg	Width	Score	Cvrg	Width	Score	Cvrg	Width	Score	Cvrg	Width	Score
Transform	Method	Model															
Shifts	k-Means	Lasso	84.7	4E36	4E36	86.5	6E6	6E6	88.6	17.4	97.1	89.2	3E3	5E4	84.9	5.48	31.3
		RF	87.4	2.41	6.79	95.5	2.18	2.27	**96.1**	2.12	2.16	95.9	2.14	2.2	96	**2.06**	**2.15**
	ϵ		0.25			0.375			0.5			0.625			0.75		
	DBSCAN	Lasso	92.8	2E134	2E134	87.3	1E30	1E30	85.4	2.31	2.85	85.8	715	716	88.9	4.16	4.86
		RF	88.9	2.39	5.33	93.9	1.91	2.24	94.5	1.92	2.06	94.8	**1.91**	**2.04**	**95.2**	1.95	2.05

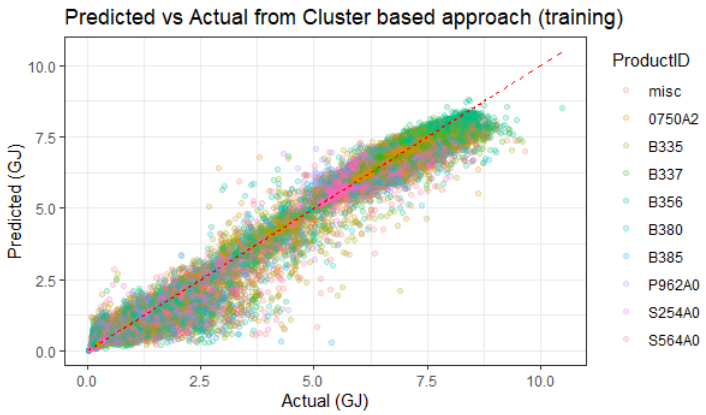
Table 5.6: Similar evaluation as in table 5.5, but for 95% PIs (constructed for the lasso and RF respectively with S-CI and QRF). Note that we only consider the shifts-based clustering here, since this gave the best results previously.

Yet another disadvantage of k-Means is its randomness, which is much less of an issue for DBSCAN. This is because its randomness is limited to the assigning of border points, which usually makes up for a very small part of the data [62]. In practice, we would therefore recommend only choosing the best clustering configuration based on *S_Dbw*, if DBSCAN is used (or similar algorithms that can differentiate noise from clusters and are mostly deterministic), if a model is considered that can adapt well (to noise) and if the corresponding visualization shows well-defined clusters / clear separation between noise and clusters.

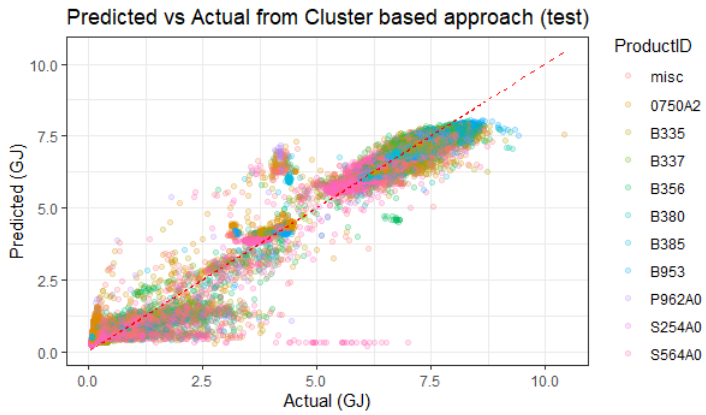
Let us now consider the best results, and compare them to the previous ones to better understand the kind of improvement that the cluster-based approach brings.

Starting with the point accuracy, the best results are obtained by using the shift-transform with DBSCAN ($\epsilon = 0.75$) and the RF model. Comparing it with the stand-alone RF model, we see an approximate improvement of 50% for the average error, which is a very significant improvement. Contrast this with the improvements that the standalone LASSO and RF models gave at the time, which were approximately 4% and 23%.

Figure 5.7 shows diagnostic plots for the cluster-approach. Comparing them to those of previous models (4.4 for OLS and 5.2 for RF), again shows that while there are some large residuals (as particularly evidenced by the heavy tails in the QQ-plots for example), errors along the bulk of the data have decreased. For example, the scatterplots between the actual and predicted values for the test data show no significant parts of the data where a structural misfit occurs.

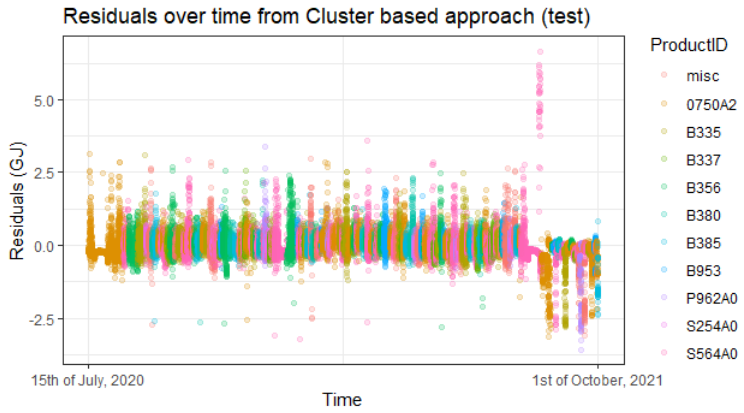


(a) Predicted values from the cluster-based approach plotted against actual values, for the training data.

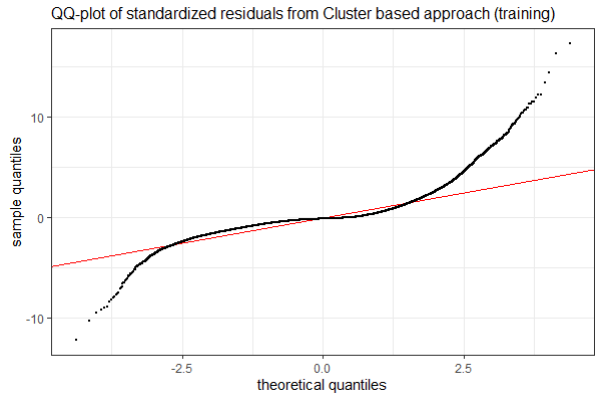


(b) Predicted values from the cluster-based approach plotted against actual values, for the test data.

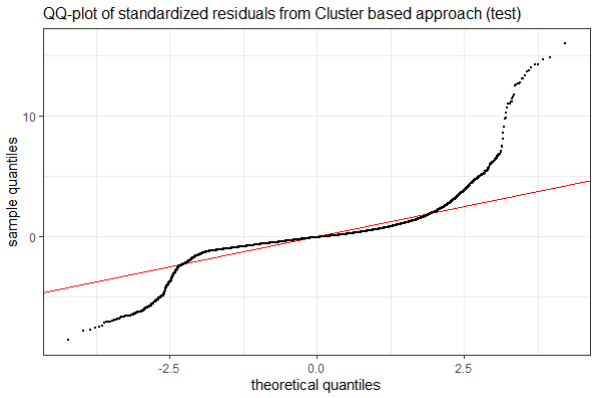
(c) Residuals over the training period for the cluster-based approach.



(d) Residuals over the test period for the cluster-based approach.



(e) Sample quantiles of the residuals from the training data are compared with the theoretical quantiles from a normal distribution, to assess normality.



(f) Sample quantiles of the residuals from the test data are compared with the theoretical quantiles from a normal distribution, to assess normality.

Figure 5.7: Diagnostic plots for the cluster-based approach. The colored points denote the top products.

For the interval accuracy, it is less clear what kind of configuration is ideal for RF, as several seem to give near identical results. Since the same DBSCAN configuration as before is among them, we can keep using the same clustering. The improvement is again very clear, with intervals that both perform better and are smaller compared with the standalone RF model. Whereas before the S-CI method together with LASSO was the only way to produce nominal PIs for the overall dataset, we now see that it is the QRF method that produces nominal PIs while also having a much smaller width. As comparison, the former models for lasso and RF had average widths of 5.57 and 2.59, and now we have reduced it to 1.95

while still retaining nominal coverage. For a visual comparison of the PIs produced by lasso and cluster-based approach, see appendix A.3.
Using the best clustering configuration for the overall data, i.e. RF with DBSCAN, we show the results for every period in 5.7, and visualize the predictive performance in figure 5.8. There is a clear improvement all around, and we now obtain a much more comparable predictive accuracy between the products. In particular, we obtain much better results for the new products and *0750A2*, which were difficult to predict for using the stand-alone models.
For the transition periods however, we unfortunately do not get the universal improvement that we were expecting, with the exception of short transitions with product switches. It still remains the case that for nearly all transition periods we do not get a nominal coverage, although the score value did still decrease for all periods. We think that this lack of improvement is because the transition periods are harder to classify into clusters, as compared with active production periods which turn out to have their own well-defined cluster (as we shall show).
Therefore, there is currently not a single approach that can provide a nominal coverage for every kind of transition, and this is still where the benefit of Lasso/S-CI and the standalone RF model lies, since only they respectively provide a nominal coverage for long and short (same product) transitions.
Furthermore, looking at the performance by month, we see that the largest improvements have occurred for August and September, which were the most troublesome months for the previous models. The predictive errors during these months is now much closer to those of other months, though they are still clearly quite bigger.

Predictive point/interval accuracy							
Type	Metrics	Period					
		Overall	Act. Production	Transition			
				Long	Medium	Short (diff)	Short (same)
Point	RMSE	0.42	0.25	0.56	0.74	0.42	0.5
	MAE	0.25	0.18	0.33	0.54	0.32	0.39
	R^2	0.98	0.89	0.5	0.8	0.92	0.92
Interval	Width	1.95	1.59	2.41	3.28	2.28	2.38
	Cvrg	95.2	99.3	81	87.9	98	89.3
	Score	2.05	1.59	2.66	3.54	2.42	3.02

Table 5.7: Predictive performance of the cluster-based approach with the RF model, using the best clustering configuration from the overall scenario.

The performance by ordered consumption groups also shows a clear overall improvement compared with the standalone RF model, particularly for the second ordered group (although it still clearly has the worst point accuracy). However, the coverage for the lowest consumption group still remains around the same as for the standalone RF model, and therefore is not nominal and remains worse than that for the stand-alone lasso model (even though all other groups do show better performance). Since we saw something similar for the RF model before, this seems to again indicate that the QRF method itself simply can not adapt well to long-term shutdown periods, since the lowest energy consumption values occur during these. This can be visually confirmed in appendix A.3, where we can see that the PIs produced by the cluster-approach more often tend to over-estimate during shut-down periods. So even though the cluster-approach has aided in performing better for nearly every group, especially in terms of point accuracy, it is still not very effective for constructing PIs for the lowest consumption values.

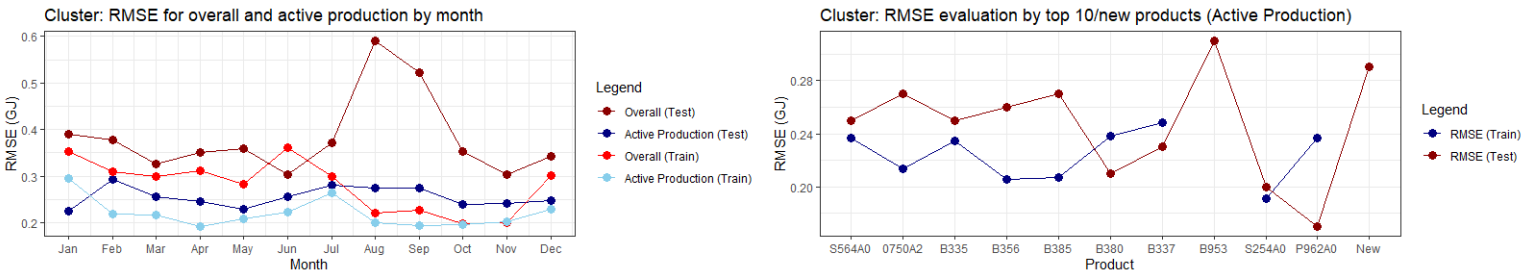


Figure 5.8

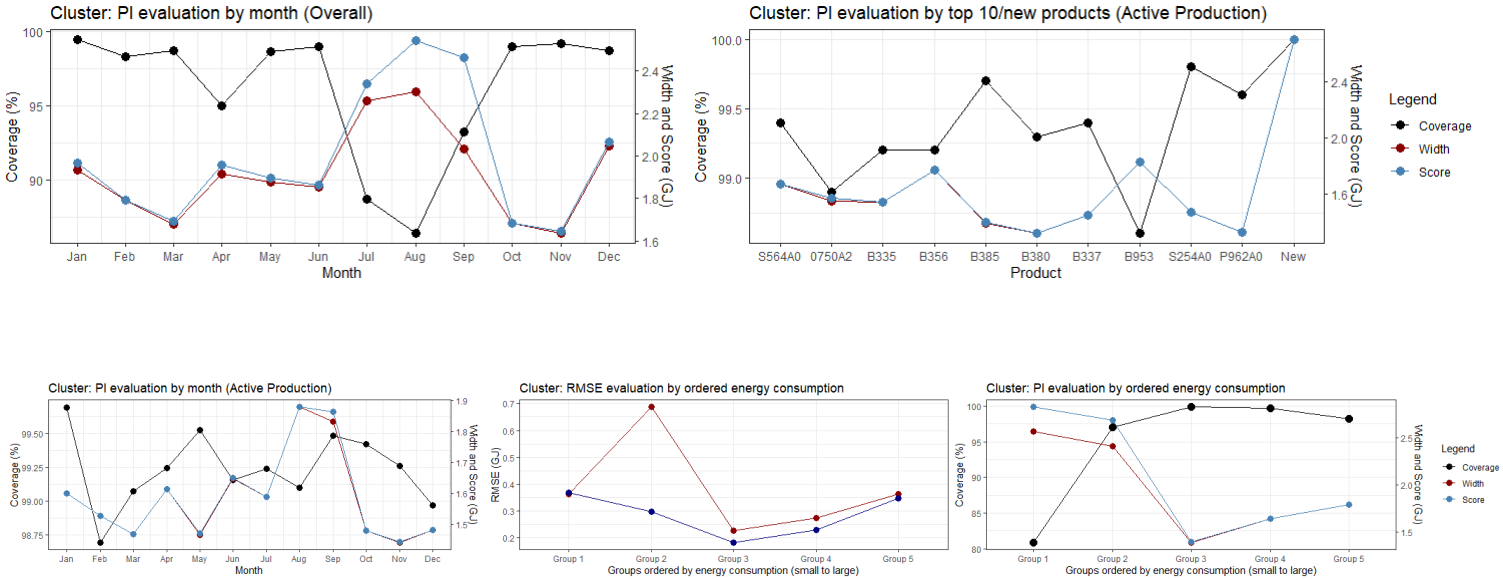
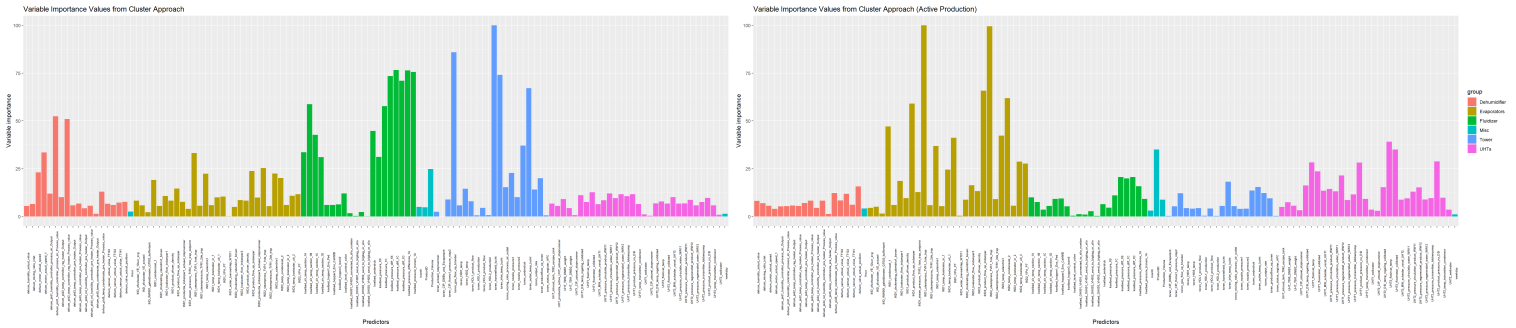


Figure 5.8: Predictive performance of the cluster based approach, shown for the top 10 products, every month and ordered energy groups.

As customary, we show the variable importances in figure 5.9. Unlike before, the cluster-based approach brings a complication in calculating the variable importances, as we require multiple fits on separate clusters to obtain the final results. Therefore, the importance values from the separate fits need to be combined in order to get a final value. In this case, we simply took their weighted average, where the weights are determined by the amount of observations in the clusters.
Unsurprisingly, the importances for the overall data appear to be approximately the same as for the standalone RF model. On the other hand, the importances during active production show larger differences compared with before, particularly for the evaporators.

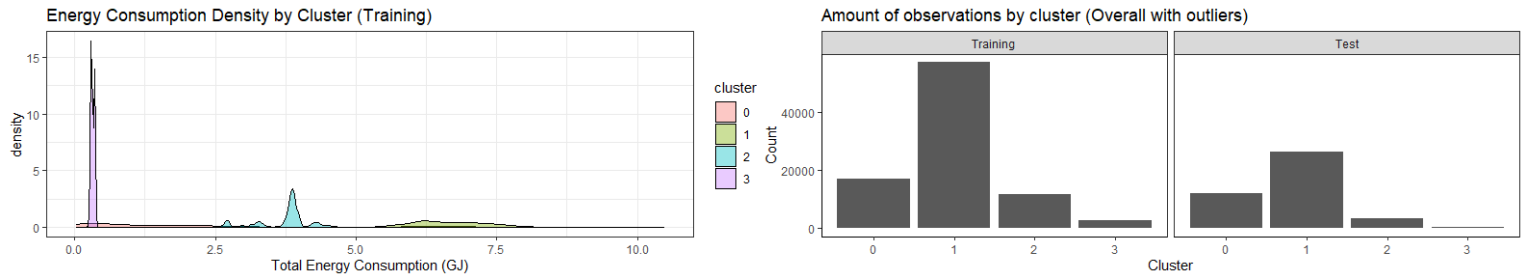


(a) Overall.

(b) Active production.

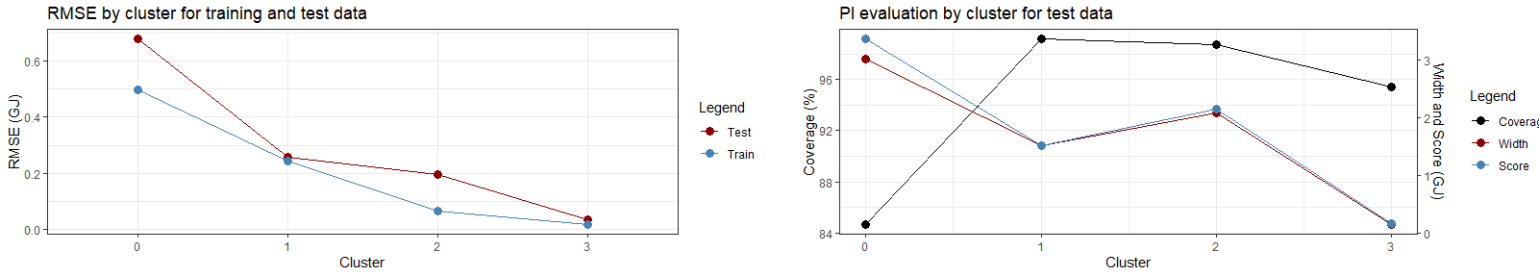
Figure 5.9: Variable importance values from the cluster-based approach for the overall situation and active production periods.

In order to yet gain further insight as to why the cluster-based approach is so effective for improving predictive performance, we try to visualize the characteristics of the formed clusters in figures 5.10, 5.11 and 5.12. The first figure shows some general properties of the clusters, namely the separate energy consumption distributions and the amount of observations. It seems that the dense clusters as found from DBSCAN denote different levels of energy consumption, with cluster 1 clearly representing the active production state. This is also supported by the other figures, in particular by the count of the tower's programme step number (similar counts for the evaporators and UHTs are shown in appendix A.4). Furthermore, as mentioned before, cluster 0 represents the noise. Clusters 2 and 3 are more ambiguous, with the former seemingly representing the boundary between active production and the rest, while the latter primarily consists of the consumption values during the shut-down periods. Regardless, the predictive performance for the non-noise clusters is shown to be very good, which explains how the cluster-method has managed to improve the predictions.



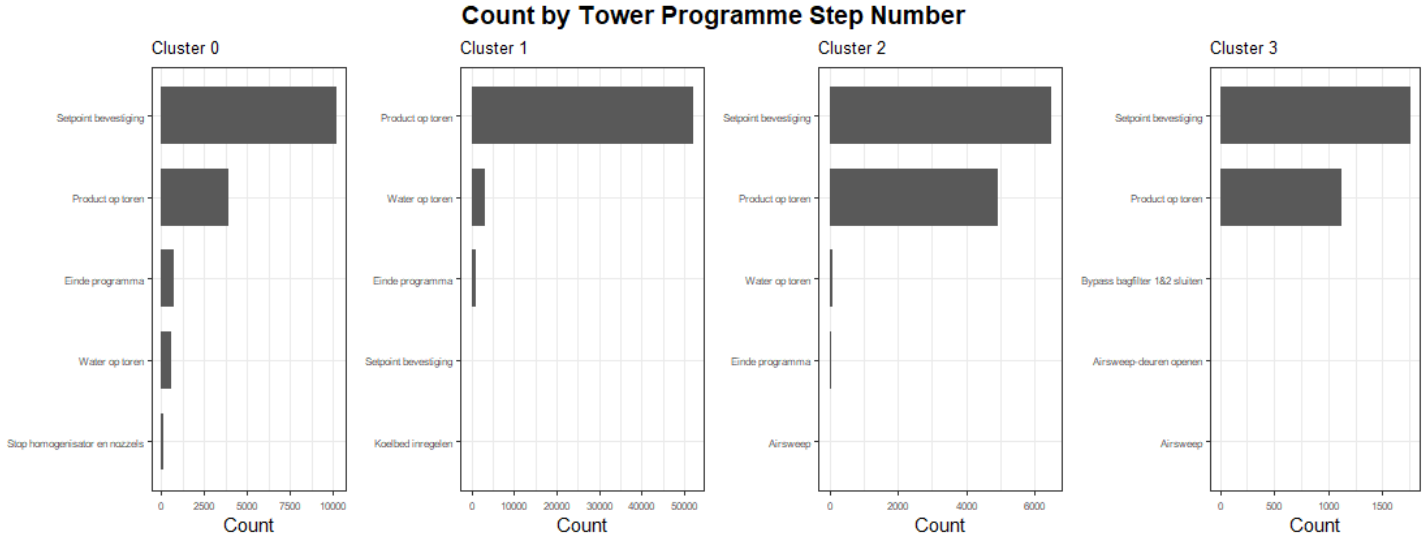
(a) (b)

Figure 5.10: The overall composition of the clusters, where in (a) we show the energy consumption densities by cluster and in (b) we show the amount of observations by cluster.



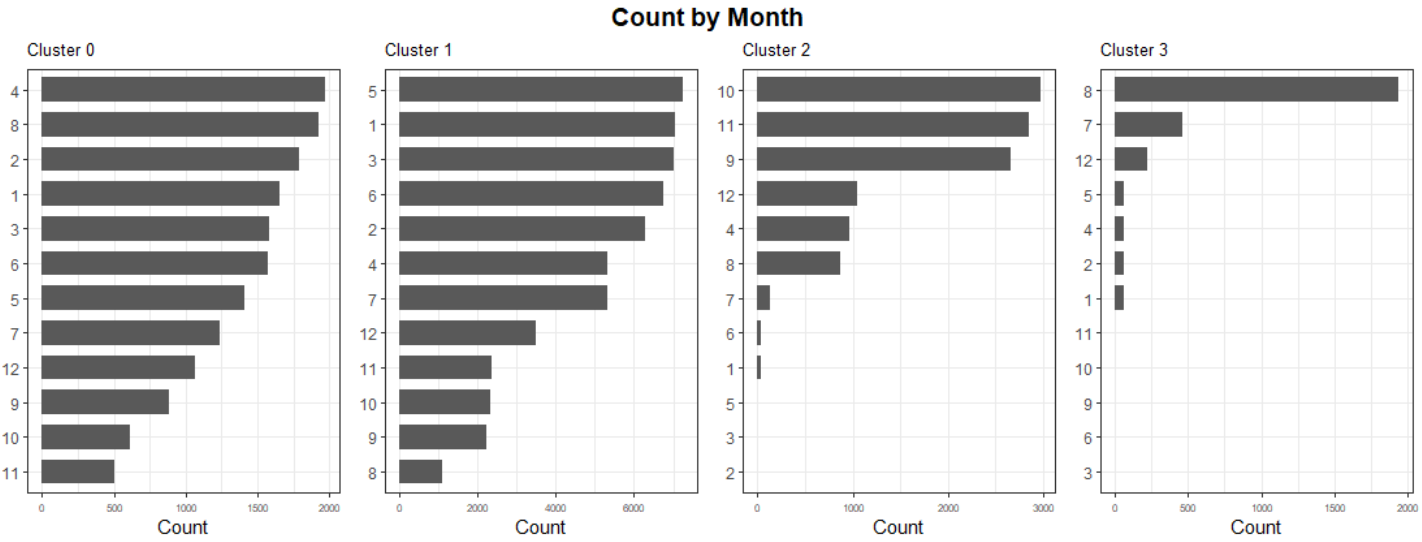
(a) (b)

Figure 5.11: Predictive performance by cluster.



(a)

Figure 5.12

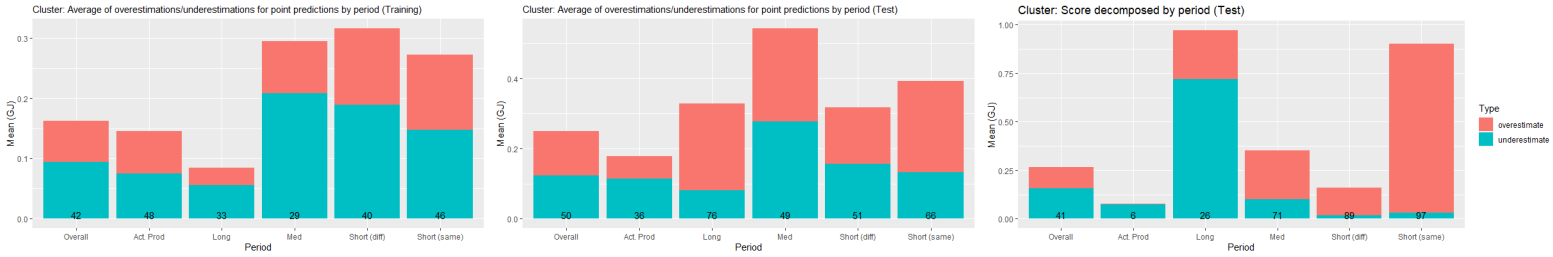


(b)

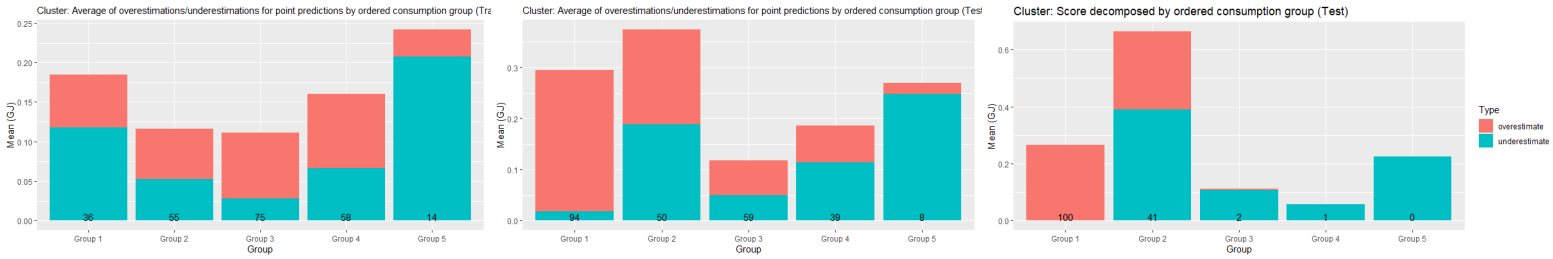
Figure 5.12: A more detailed visualization of the composition per cluster, where in (a) we show the counts for the tower programme step numbers, and in (b) we show the counts for every month.

5.3. Application

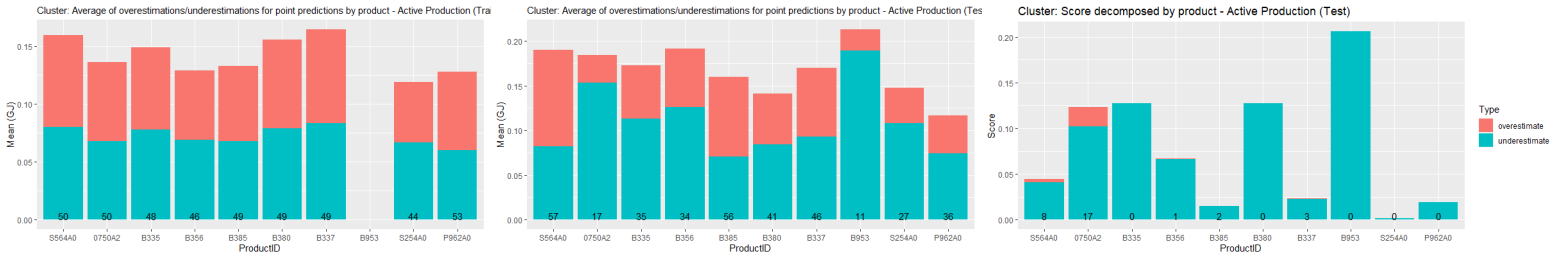
As a demonstration of how predictive models can be used to assess whenever the largest changes in energy efficiency occur, we will use the currently most accurate predictive model (cluster-based approach) to assess the balance between over-and underestimations, and see where the largest changes in this balance have occurred between the training and test data. Of course, this should be conditional on whether we see a good balance in the training data (i.e. the contributions should be approximately between 40% and 60%), as this is an indication that the model fits well and therefore provides a good energy baseline.
We visualize some examples in figure 5.13, where we show the contributions for both the training and test period as measured by the MAE metric, and for the latter period we also show the contributions from the score evaluation metric as we already described in section 4.1.
Looking at the first two examples, which show the balances for each of the considered periods and the ordered energy consumption groups, we can note some interesting results. For the overall situation, it would seem that the contribution from overestimations has increased, and therefore that seemingly the overall energy efficiency has likewise improved. However, since this improvement is largely made up by the improvements for the long and medium transition periods, which in turn show clear imbalances for the training data, it is not yet clear if this "improvement" is not simply caused by the model's poor adaptability for these periods. A more safer conclusion can be made by looking at the active production periods, for which the model does adapt well. In this case, a clear decrease in overestimations is visible, which means that during these periods the efficiency has decreased. This conclusion is also supported by the results for the ordered groups, since we see a similar decrease for the highest consumption groups. It is also still interesting to note that we see such a large increase in overestimations for the lowest consumption group and long-term shut-down periods, indicating that adaptability issues non-withstanding, the efficiency may have improved during such periods.
Focusing on the results for the top 10 products, we indeed see that most balances in the training data are well-balanced, again indicating the good performance of the model during active production. In terms of products, we see that the largest decreases in efficiency occur for *0750A2* and *B953*, which after confirmation from a process engineer appear to make sense, since these products require relatively more energy. Especially the result for *B953* is noteworthy since this was a new product (and that the model had not been trained on), and therefore this indicates that the current approach could be quickly used to assess the efficiency aspect for completely new products. The only products that do not show a decrease in efficiency are *S564A0* and *B385*.
In terms of months, for the overall scenario we see that decreases in efficiency have occurred along the beginning of the year (January to April) and October, and increases in efficiency have mainly occurred along the middle portion of the year (May to September) with the addition of December. November seems to be the only month without a worthwhile change. Also, the results for active production again confirm what we saw before, with clear decreases in efficiency for most months (with the exceptions of May, August and September).



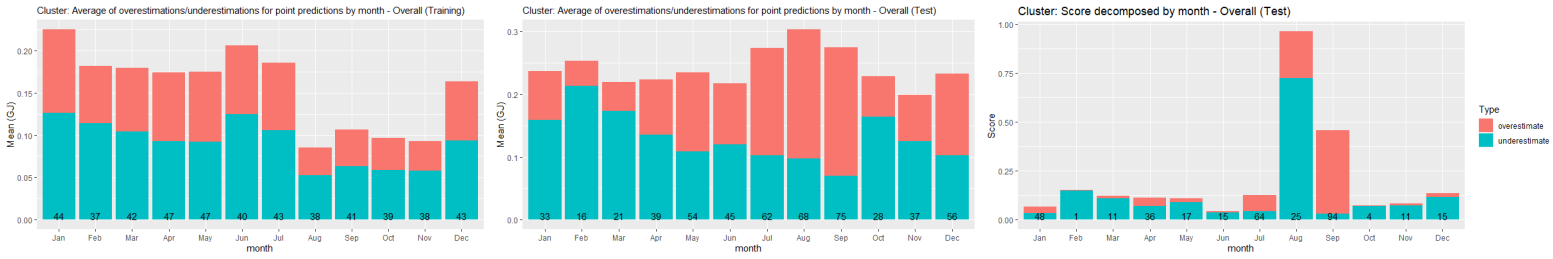
Contributions by period.



Contributions by ordered energy consumption groups.



Contributions by the top 10 products.



Contributions by month for the overall scenario.

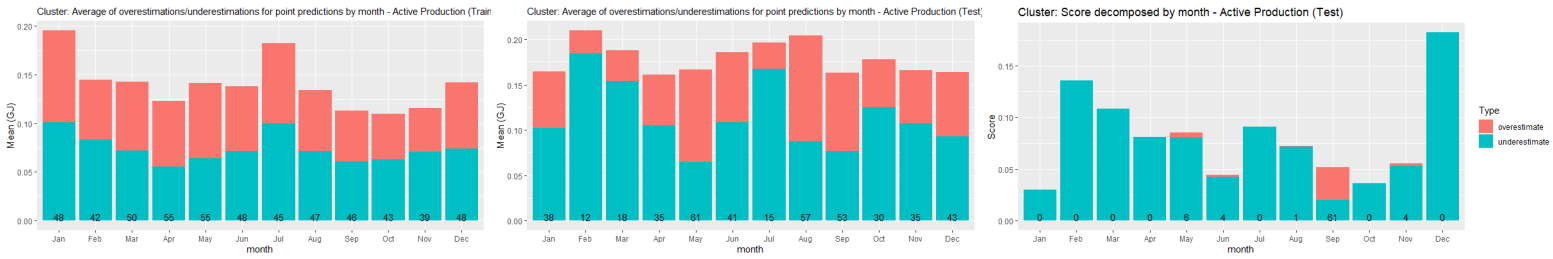


Figure 5.13: Contributions by month for active production periods.

Finally, we show the top 10 variables for the three most accurate models (MARS, RF and cluster-model) in figure 5.14. As explained before, to find the most important variables we are particularly interested in those variables that are important for every model, which in this case are 3 variables directly related with the tower: amount of air in the tower, and the inlet/outlet temperature of the drying air in the tower. An additional confirmation of their importance can be seen from the biplot we made from the PLS model in figure 4.14a, where these variables clearly have some of the longest arrow lengths (indicating their importance) and their directions are in approximately the same direction as that of the total energy (indicating their correlation with the output variable).

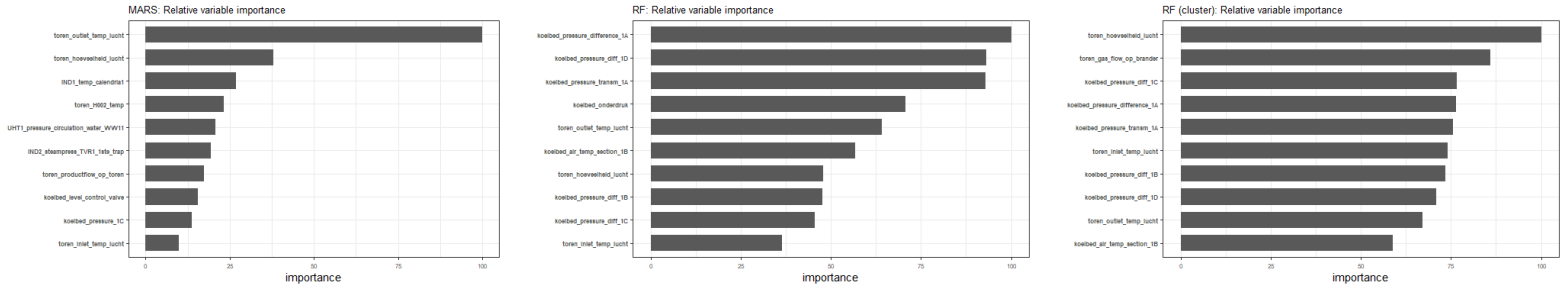


Figure 5.14

6

Summary and recommendations

In this thesis we have focused on performing a statistical analysis of the total energy consumption of the powder drying process, as performed in Abbott's drying plant located in Zwolle. The primary motivation behind this project was that the drying process consumes a lot of energy, with it in fact accounting for approximately 70% of the total consumed energy in the entire plant. Our main task therefore was to find accurate predictive models for the total energy, and to additionally demonstrate how such predictive models could be used to gain further insight into where the largest changes in energy efficiency had taken place and the dependencies of the total energy consumption.

Our first step was data-exploration, where we made some general visualizations of the energy consumption over the test period. This allowed us to better understand how the data looked like, and to make the necessary adjustments for the subsequent steps (for example, which top products we had to focus on). We also performed a preliminary cluster analysis on the daily energy profiles, where we found that the data could be split into distinct groups. This served as a first prior justification for our cluster-based approach later on, which was our novel predictive model where we fitted a separate model on each found cluster.
In order to provide a good picture of the predictive accuracy, we had focused on two forms to provide the predictions in, i.e. prediction *points* and 95% *intervals*. Furthermore we considered a good variety of different periods, which we broadly categorized as the overall, active production and transitional (long, medium, short (different product), short (same product)) scenarios.
To start with our predictive modelling, we first focused on linear regression models. We considered OLS, the regularized variants (ridge, lasso, elastic net) and dimension reduction models (PCR, PLS). Linear regression generally serves as a good stepping stone due to its simplicity, and would therefore serve as a reference model throughout the study (in particular through the lasso model). The considered linear regression models aside from OLS are generally known as complexity reducing models, due to their ability to automatically discard irrelevant variables and obtain a smaller list of variables. We found that only lasso and elastic net were capable of improving predictions, if only slightly (since these models gave identical predictive accuracy, we henceforth ignored elastic net since it was more difficult to fit). For the PIs we also compared different methods. These were the analytic, bootstrap, quantile regression, split conformal inference (S-CI) and multi S-CI approaches, with the latter 4 being computational in nature. By simultaneously considering the evaluation metrics and computational efficiency, it was found that lasso together with S-CI gave the best PIs at this point.
Using the PLS model, we also considered a second cluster analysis using all the variables at once, instead of solely the output variable as we did during data exploration. Again, we found evidence that the data could be split into distinct groups, giving another prior justification for our cluster-based approach.
To better figure out how the dependencies looked like we made several visualizations, which were all separated into 5 distinct groups of the drying process: Evaporators, UHTs, Tower, Fluidizer and Dehumidifier. In particular we showed the variable importance plots for the lasso model colored by each group, and using the PLS model we made a bi-plot showing the correlation structure between the variables along with their importances. The main conclusion was that the tower-related variables consistently showed up as an important group, and that there were clear groups of well-correlated input variables.

The second set of models that we considered were the non-linear MARS and RF (Random Forests) models, since they represented natural steps in complexity. In particular, MARS could be regarded as a combination of linear regression and non-linear modelling, while RF was a fully non-parametric approach that focused exclusively on the data. In terms of point accuracy we had found that they could unsurprisingly perform quite a bit better, although this was not necessarily

the case for the intervals. The main issue was under-coverage, meaning that often the coverage rates were well below the nominal value of 95%. We also made similar plots as before to visualize the dependencies, and even though some groups of variables swapped in importance, the tower-related group still had a large average importance.
The final predictive approach that we considered was the cluster-based approach, where we would cluster the data in several ways. The idea was that each found cluster would contain some specific patterns, and a separate model fit to each such cluster would be better able to model those patterns. To form the clusters we compared several clustering configurations, where we considered several clustering algorithms (k-Means, hierarchical, DBSCAN) and several ways to transform the original data into a lower dimensional space to visualize the clusters (PLS or using 8 hour-shift profiles together with the first two principal components). The final clustering configuration was chosen by considering both the optimal value for a CVI (cluster validation index) and a visualization of the clusters, where we were mainly interested in dense and well-separated clusters.
Implementing the cluster-based approach in this way together with RF to perform the separate cluster fits, ended up giving a huge improvement to predictive accuracy as we summarize below in tables 6.1a and 6.1b. In nearly every scenario, the cluster-based approach gave a large improvement compared with the standalone RF model, let alone the linear regression models. What is also noteworthy is that this improvement extends to the new products and interval accuracy, with in particular the coverage being much closer to nominal. However, as with the stand-alone RF model, there were still some scenarios were the coverage was lower than with lasso, in particular for the long transition periods.
We then made a quick demonstration of how the most accurate predictive model could be used to find the largest changes in energy efficiency. Essentially, the model represents the training period, and therefore its predictions serve as an energy baseline for the test period. Therefore by considering the balance in contributions of over-and underestimations, we can regard the largest imbalances as primary indicators for the largest changes in efficiency, i.e. the efficiency has likely decreased if the overestimations have a small contribution and vice-versa. An additional requirement that we had made to make these conclusions, was that the corresponding contributions for the training data needed to be balanced, as this would indicate that the model's fit was not the cause.
Finally, in order to gain further insight in the dependencies, we summarized the top 10 variables for the three most accurate models and found that three variables were consistently important (which were all tower-related).

Summary of Predictive Point Accuracy				
Period	Model			
	OLS	LASSO	RF	Cluster
Overall	1.22	1.17 (4.1)	0.89 (27)	**0.42** (**65.6**)
Act. Prod	0.74	0.69 (6.8)	0.53 (28.4)	**0.25** (**66.2**)
Long	0.62	0.61 (1.6)	0.59 (4.8)	**0.56**(**9.7**)
Medium	1.08	1.07 (0.93)	**0.72**(**33.3**)	0.74 (31.5)
Short (diff)	2.23	2.24 (-0.45)	1.03 (53.8)	**0.42** (**81.2**)
Short (same)	1.85	1.92 (-3.78)	1.03 (44.3)	**0.5** (**73**)

(a) RMSE for several models.

Summary of Predictive Interval Accuracy				
Period	Model			
	OLS	LASSO	RF	Cluster
Overall	4.41	7.18 (-62.8)	6.83 (-54.9)	**2.05** (**53.5**)
Act. Prod	3.7	6.28 (-69.7)	4.04 (-9.2)	**1.59** (**57**)
Long	3.05	3.83 (-25.6)	3.02 (0.98)	**2.66** (**12.8**)
Medium	4.16	7.03 (-69)	3.8 (8.65)	**3.54** (**14.9**)
Short (diff)	8.63	18 (-109)	7.02 (18.7)	**2.42** (**72**)
Short (same)	5.68	11.4 (-100)	5.07 (10.7)	**3.02** (**46.8**)

(b) Score for several models.

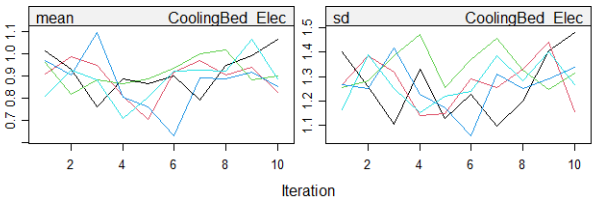
Table 6.1: Predictive performance summarized for the models considered at each stage. The value in parentheses indicates the percentage change compared with OLS.

The first main recommendation that we can make, is that in further applications of our described procedure for finding the largest changes in energy efficiency, the training and test data should be more carefully chosen. Our focus in this project was to develop a general method only and demonstrate it for an arbitrarily chosen partition of the data, and therefore we did not focus on any specific periods. Of course, for a proper application of the method, the training data in particular should not contain any structural changes in efficiency, in order for the fitted prediction model to return a proper energy baseline. In our case, the non-adherence to this might have caused some of the imbalances we saw in the training data for the cluster-method, when we were evaluating where the largest changes in efficiency had occurred.
Furthermore, in order to confirm the consistency of the results, we recommend that the analysis should be repeated by considering other accurate predictive models, and by measuring the change in efficiency through alternative methods. An interesting alternative for the latter, could be to predict all 99 quantiles from the first to the 99th, and to see between which quantiles the actual value lies.
It is also especially recommended that other approaches for constructing prediction intervals are considered, because of all the approaches considered in this project, there is still not a single one that provides the best results in every situation. For example, the generally more accurate non-linear models and cluster-approach did not provide a better coverage than lasso during long-term shut-down periods.

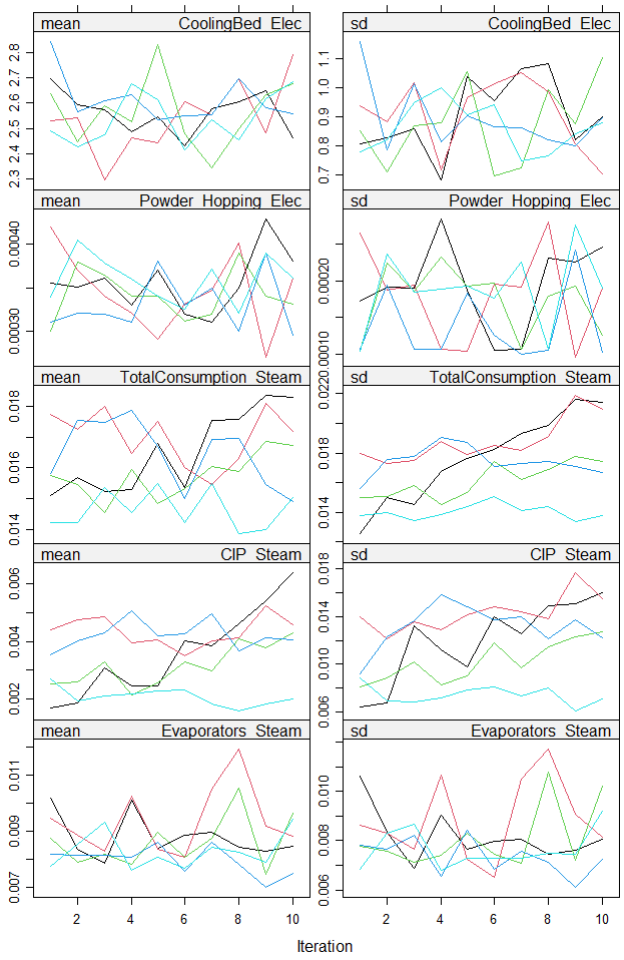
A

Appendix A

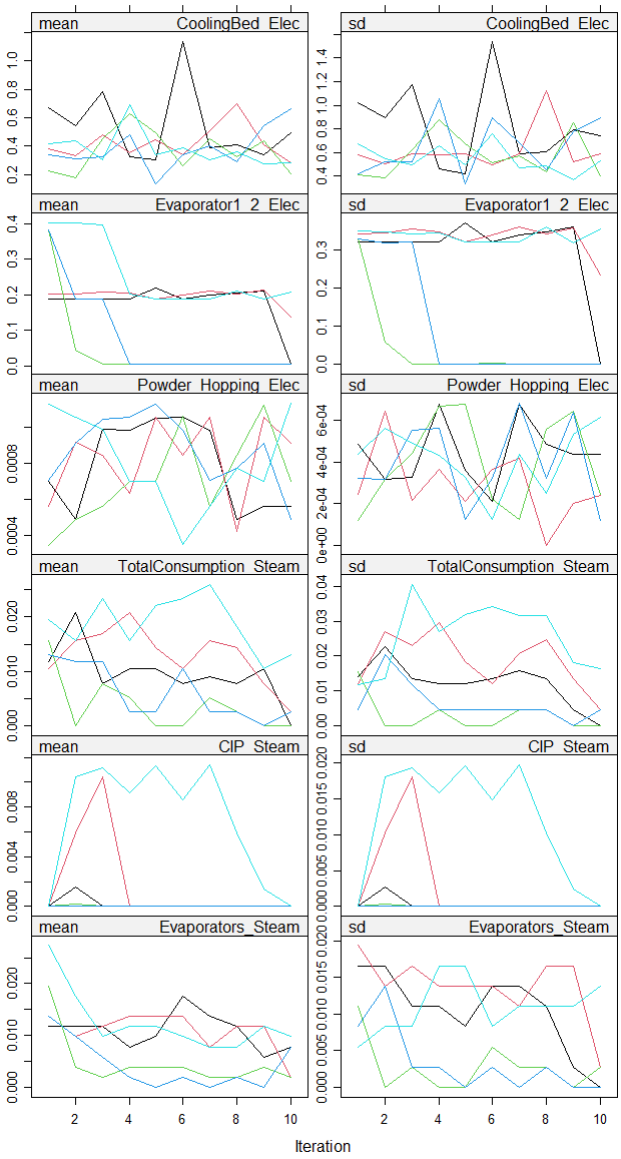
A.1. Traces of Missing Value Imputations in Aggregated Data



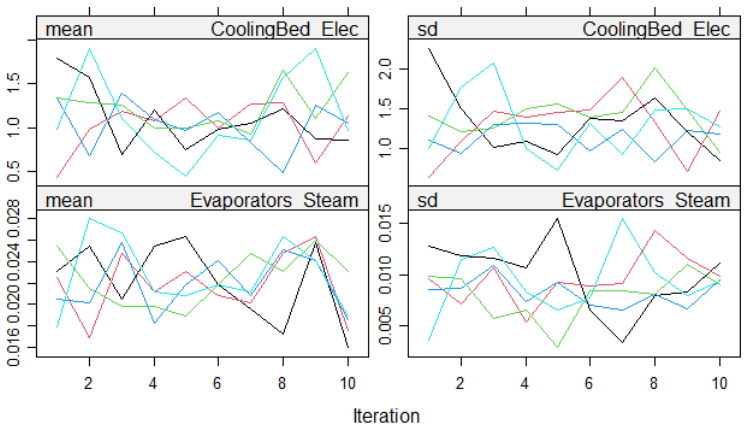
Set 1



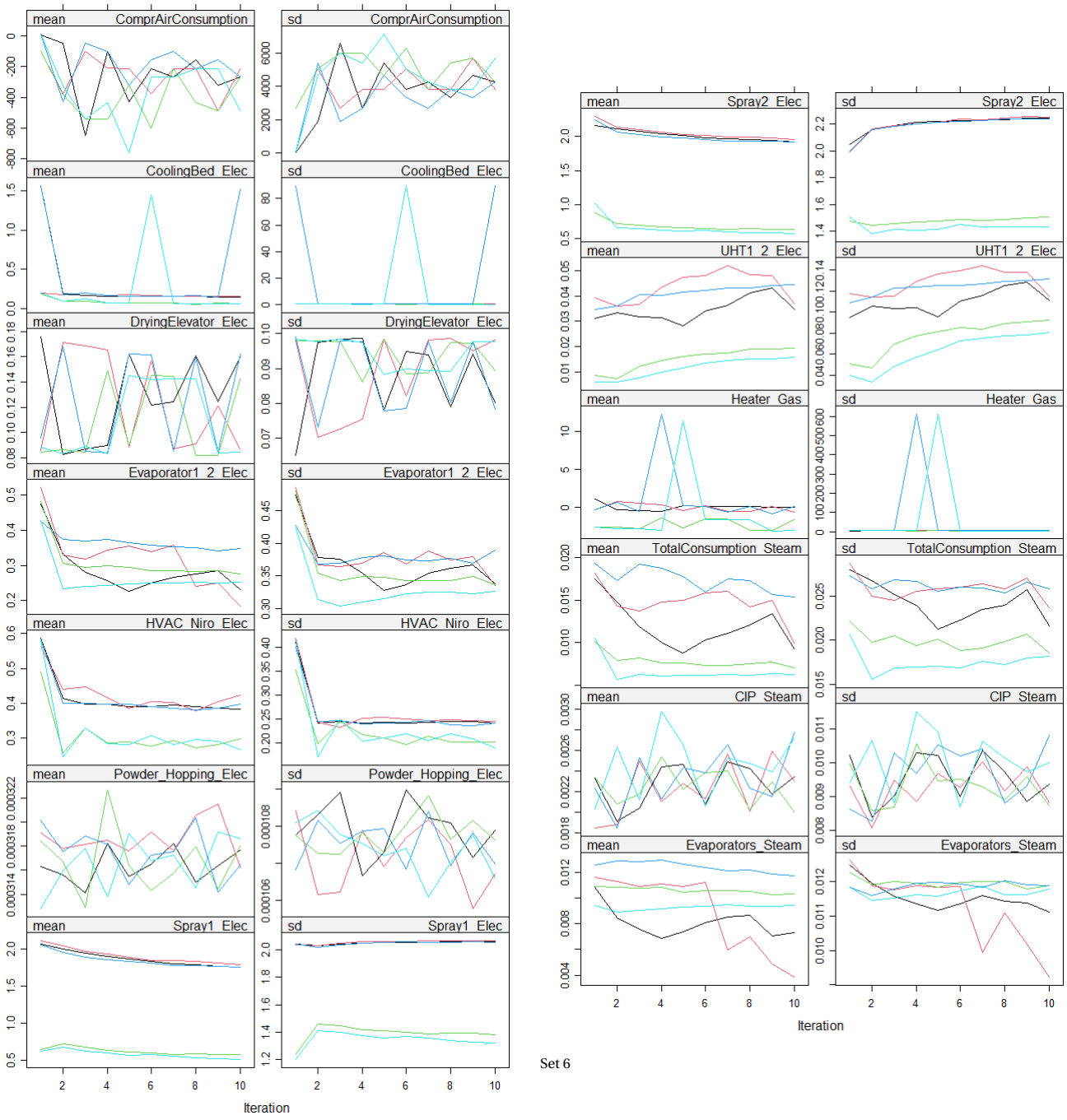
Set 2


mean CoolingBed Elec
sd CoolingBed Elec
mean Evaporator1 2 Elec
sd Evaporator1 2 Elec
mean Powder Hopping Elec
sd Powder Hopping Elec
mean TotalConsumption Steam
sd TotalConsumption Steam
mean CIP Steam
sd CIP Steam
mean Evaporators Steam
sd Evaporators Steam
Iteration

Set 3

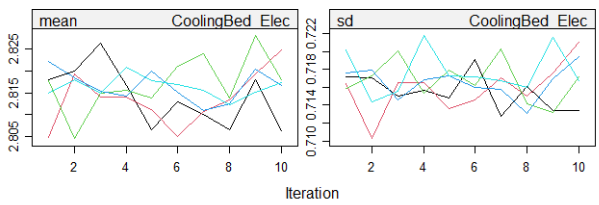

mean CoolingBed Elec
sd CoolingBed Elec
mean Evaporators Steam
sd Evaporators Steam
Iteration

Set 4



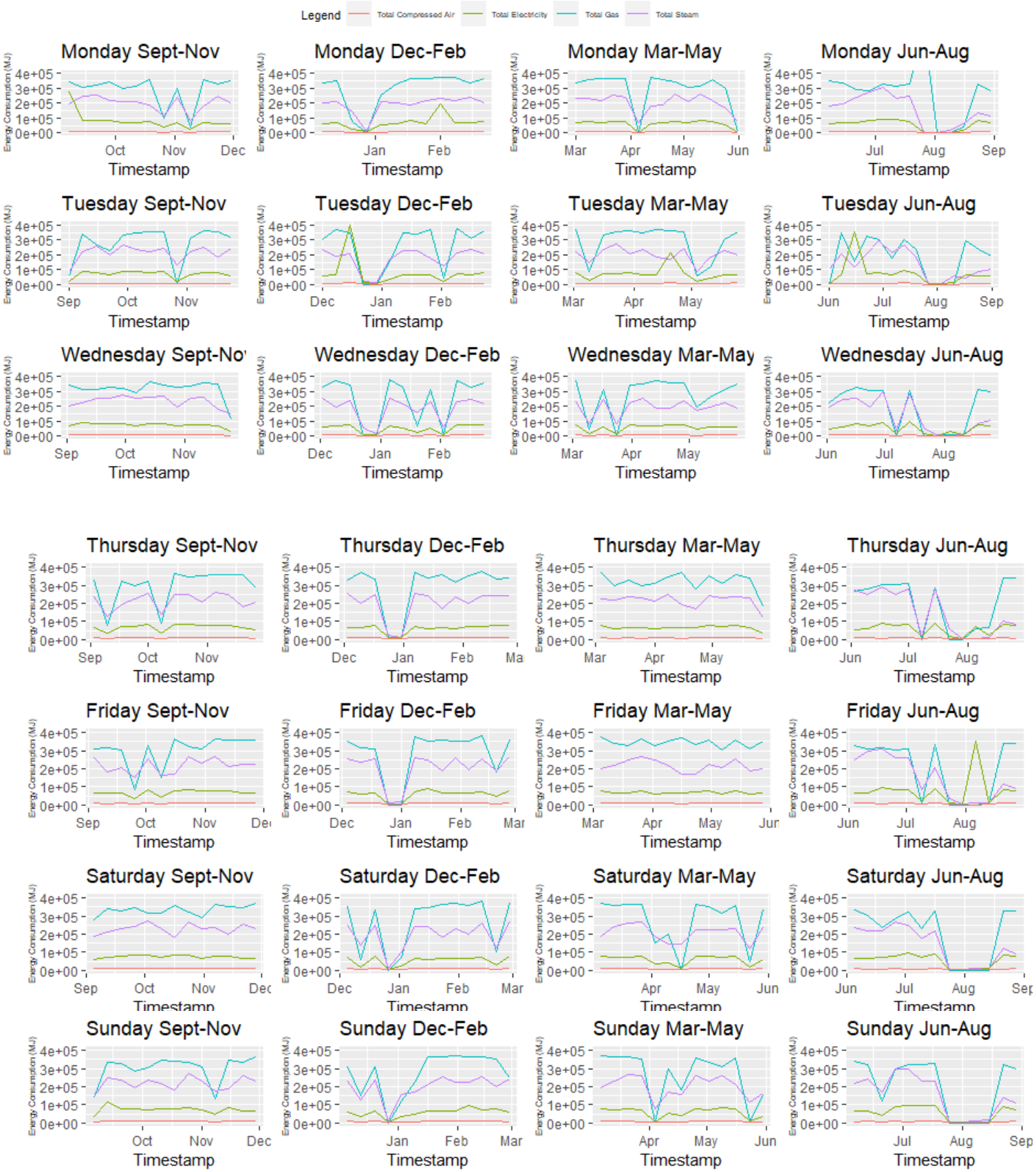
Set 6

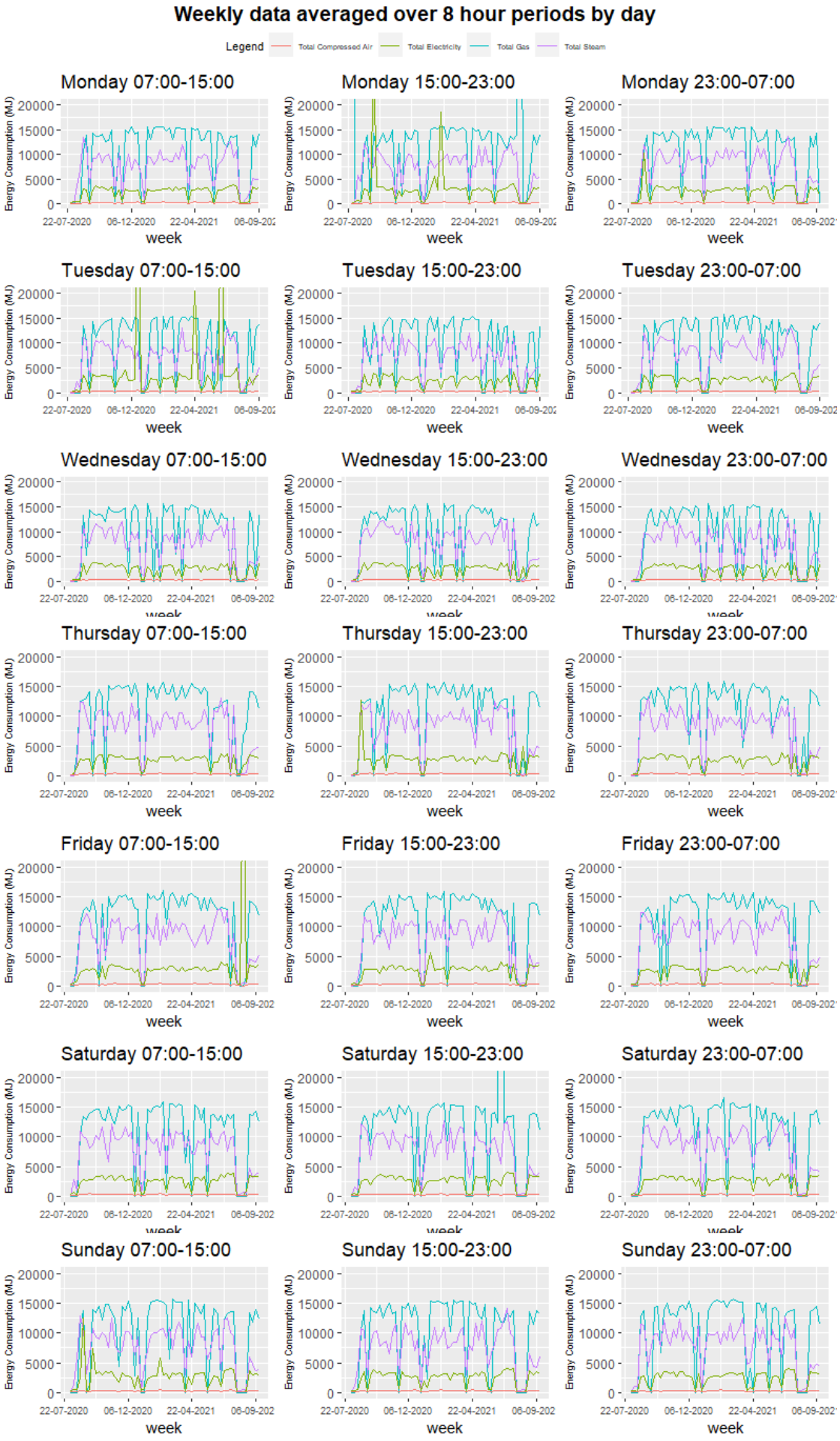
Set 6

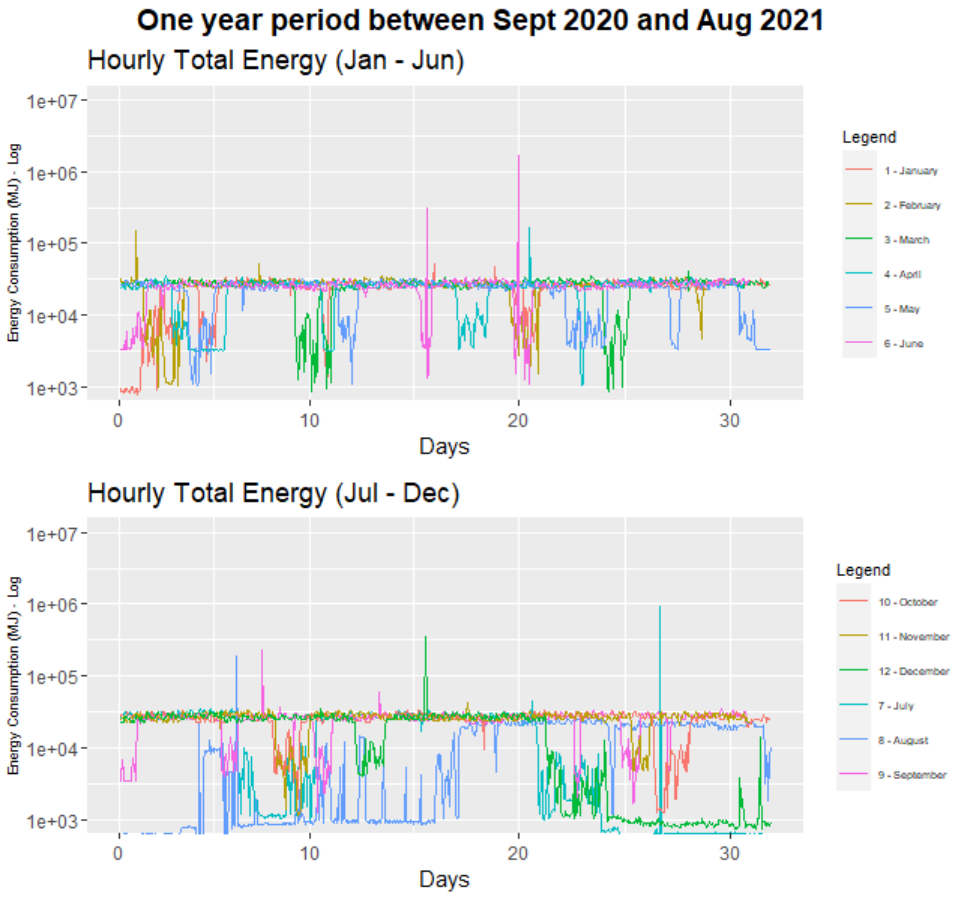


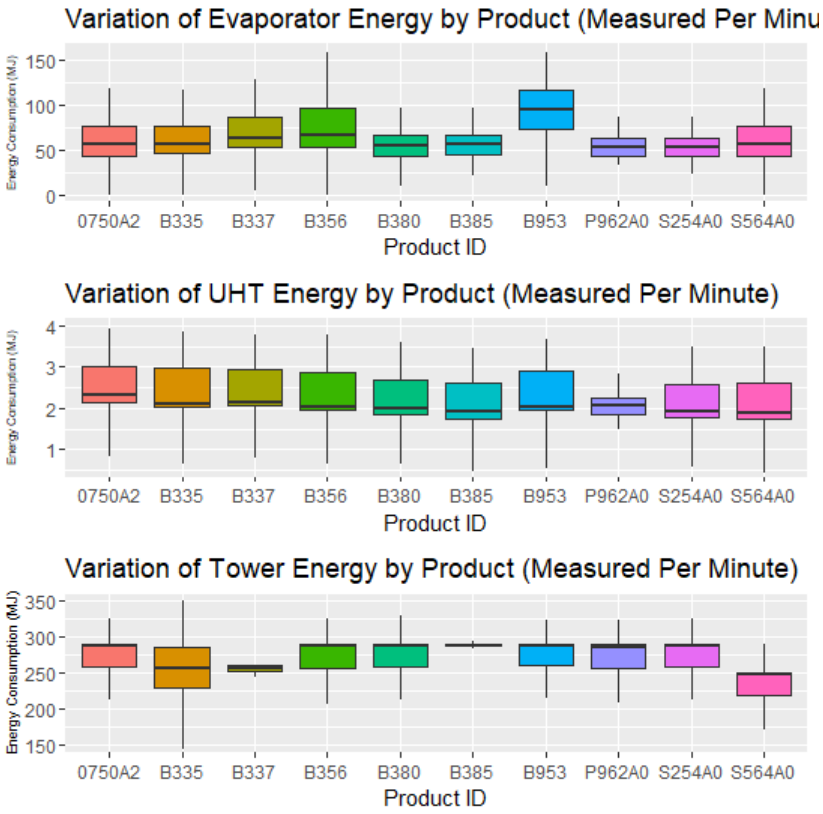
Set 7

A.2. Additional Data Visualization Figures



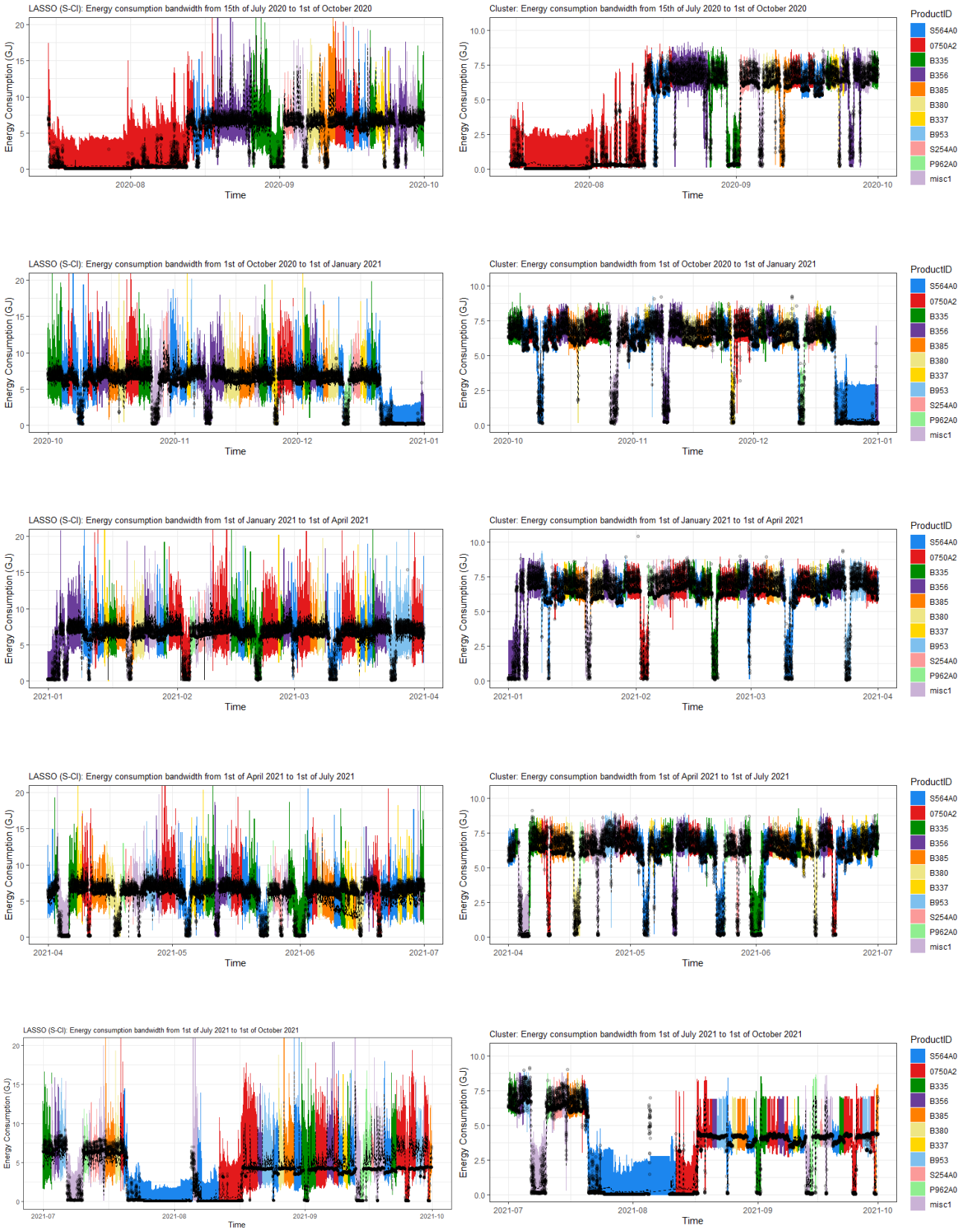

Weekly data averaged over 8 hour periods by day
Legend
Total Compressed Air
Total Electricity
Total Gas
Total Steam
Monday 07:00-15:00
Monday 15:00-23:00
Monday 23:00-07:00
Tuesday 07:00-15:00
Tuesday 15:00-23:00
Tuesday 23:00-07:00
Wednesday 07:00-15:00
Wednesday 15:00-23:00
Wednesday 23:00-07:00
Thursday 07:00-15:00
Thursday 15:00-23:00
Thursday 23:00-07:00
Friday 07:00-15:00
Friday 15:00-23:00
Friday 23:00-07:00
Saturday 07:00-15:00
Saturday 15:00-23:00
Saturday 23:00-07:00
Sunday 07:00-15:00
Sunday 15:00-23:00
Sunday 23:00-07:00
Energy Consumption (MJ)
0
5000
10000
15000
20000
22-07-2020
06-12-2020
22-04-2021
week


One year period between Sept 2020 and Aug 2021
Hourly Total Energy (Jan - Jun)
Energy Consumption (MJ) - Log
1e+07
1e+06
1e+05
1e+04
1e+03
0
10
20
30
Days
Legend
1 - January
2 - February
3 - March
4 - April
5 - May
6 - June
Hourly Total Energy (Jul - Dec)
Energy Consumption (MJ) - Log
1e+07
1e+06
1e+05
1e+04
1e+03
0
10
20
30
Days
Legend
10 - October
11 - November
12 - December
7 - July
8 - August
9 - September

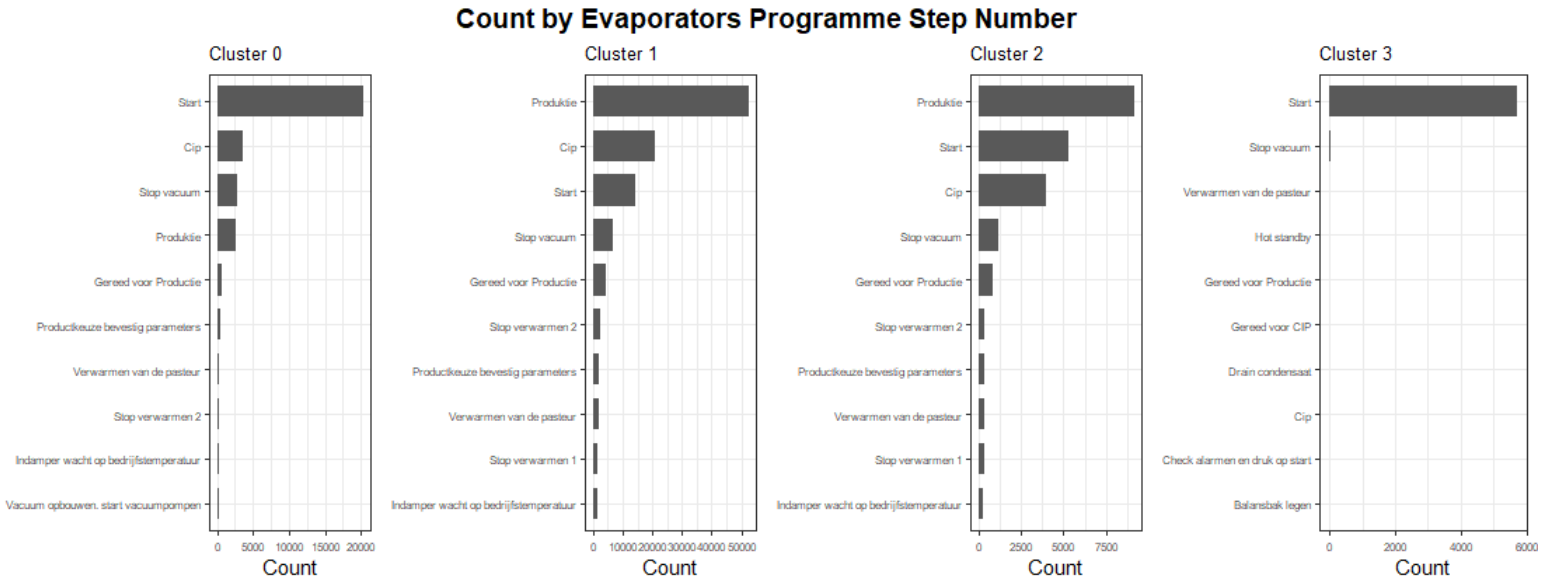

Variation of Evaporator Energy by Product (Measured Per Minu
Energy Consumption (MJ)
150
100
50
0
0750A2
B335
B337
B356
B380
B385
B953
P962A0
S254A0
S564A0
Product ID
Variation of UHT Energy by Product (Measured Per Minute)
Energy Consumption (MJ)
4
3
2
1
0750A2
B335
B337
B356
B380
B385
B953
P962A0
S254A0
S564A0
Product ID
Variation of Tower Energy by Product (Measured Per Minute)
Energy Consumption (MJ)
350
300
250
200
150
0750A2
B335
B337
B356
B380
B385
B953
P962A0
S254A0
S564A0
Product ID

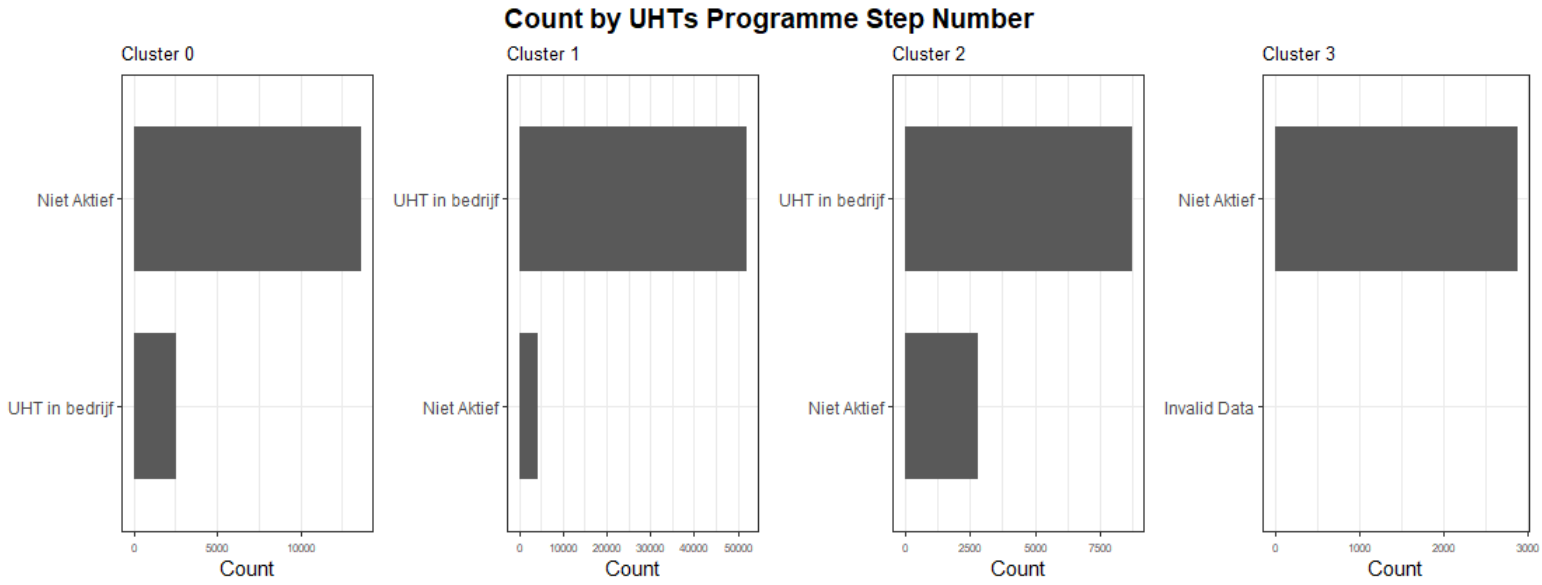
A.3. Additional PI visualizations

LASSO **Cluster-based Approach**



A.4. Additional cluster composition visualizations





B

Appendix B

B.1. Kruskal-Wallis rank sum test

The Kruskal-Wallis rank sum test [44] tests the null hypothesis that the medians from the different groups are equal to each other. It first combines the data from all groups, and then sorts them in ascending order. Each value is then assigned a rank, after which the ranks of each group are added together and the following test statistic is calculated:

$$KW = \left[\frac{12}{N(N+1)} \sum_{j=1}^{c} \frac{T_j^2}{N_j} \right] - 3(N+1), \tag{B.1}$$

where N is the total sample size, c is the number of groups, T_j is the sum of ranks in the j^{th} group and N_j is the corresponding group sample size. Under the assumption that KW has a chi-square distribution with $c-1$ degrees of freedom under the null hypothesis, the null hypothesis is rejected when KW exceeds the corresponding critical value.

B.2. Clustering Algorithms

B.2.1. Partitional Algorithms

Partitional algorithms try to minimize the within-cluster scatter, which is represented by the following optimization problem [36]:

$$\min_{C,\{m_k\}_1^K} \sum_{k=1}^{K} N_k \sum_{C(i)=k} \|x_i - m_k\|_2, \tag{B.2}$$

where C is the cluster mapping which maps observations to clusters, m_k are the cluster centroids and N_k are the cluster sizes.
In turn, the K-means and K-medoids algorithms are described in Algorithm B.1 and B.2. Note that when using medians as centroids, the algorithm is analogous to K-Means aside from replacing cluster means with cluster medians.

Algorithm B.1 *K-Means*

---

1. For a random initial cluster assignment C and specified amount of clusters K, B.2 is minimized for $\{m_1, ..., m_k\}$ by setting the m_k equal to the cluster means $\bar{x}_k$, as for cluster k we have that $\bar{x}_k = \underset{m}{\operatorname{argmin}} \sum_{C(i)=k} \|x_i - m\|_2$.
2. With a new set of cluster means, B.2 is minimized by defining a new cluster mapping:

$$C(i) = \underset{1 \le k \le K}{\operatorname{argmin}} \|x_i - m_k\|_2, \tag{B.3}$$

   i.e. assigning observations to their closest cluster means.
3. Previous steps are alternated until $C(i)$ does not change $\forall i$.

Algorithm B.2 *K-Medoids*

1. For a random initial cluster assignment C and specified amount of clusters K, the cluster centroid m_k for cluster k is found by finding the observation index i_k^* such that:
$$i_k^* = \underset{i:C(i)=k}{\operatorname{argmin}} \sum_{C(i')=k} d(x_i, x_{i'}), \tag{B.4}$$
thus $m_k = x_{i_k^*}$ for $k = 1, ..., K$.
2. With a new set of cluster medoids, a new cluster mapping is defined as:
$$C(i) = \underset{1\le k\le K}{\operatorname{argmin}} d(x_i, m_k), \tag{B.5}$$
i.e. assigning observations to their closest cluster medoids.
3. Previous steps are alternated until $C(i)$ does not change $\forall i$.

B.2.2. Hierarchical algorithm

Agglomerative hierarchical algorithms use inter-cluster dissimilarity measures (linkages) to merge the least dissimilar clusters, starting from an initial configuration in which each observation is a singleton cluster. We summarize some commonly used linkages $L_{ij} = L(C_i, C_j)$ in table B.1, and Algorithm B.3 summarizes the algorithm. The clustering for a specified amount k of clusterings is provided by cutting the corresponding hierarchical tree at the right height, see [36].

Linkage	Complete	Average
Formula	$L_{ij} = \max_{x\in C_i, y\in C_j} d(x,y)$	$L_{ij} = \frac{1}{\lvert C_i\rvert\cdot\lvert C_j\rvert} \sum_{x\in C_i}\sum_{y\in C_j} d(x,y)$

Table B.1: Linkages used for hierarchical clustering.

Algorithm B.3 *Agglomerative Hierarchical Clustering*

1. Initialize all observations x_j as separate clusters.
2. Calculate pair-wise linkage values $L_{ij} \forall 1 \le i, j \le T$, where T is the current amount of clusters.
3. Merge the clusters C_k, C_l such that:
$$L_{kl} = \min_{1\le i,j\le T} L_{ij}, \tag{B.6}$$
and set $T = T - 1$.
4. Repeat previous two steps until only one cluster is left.

B.2.3. Density-based algorithm

Density-based algorithms, in particular "DBSCAN" [35] [26] [62], employ a more localized approach to clustering. It considers each point separately, and uses the user-defined parameters ϵ and *minPnts* to classify it as a *core*-point, *border*-point or noise.

The first parameter ϵ denotes the maximum radius that a cluster may take (aside from noise), while the second parameter *minPnts* denotes the minimum amount of observations that a cluster must contain. In the absence of domain knowledge, these parameters are obtained through heuristic procedures as described in Sander et al (1998) [62].

The separate classifications denote how "dense" a particular point p is, which is measured by how many points its *eps*-neighbourhood $N_\epsilon(p)$ contains and whether it is directly reachable from a core-point. More formally, a core-point p is a point such that $\mid N_\epsilon(p) \mid \ge minPnts$, and a point q is said to be directly reachable from p if it is contained in its neighbourhood. A point p is then defined as a border-point if it is directly reachable from a core-point q and $\mid N_\epsilon(p) \mid < minPnts$. Finally, points that are neither core-points or border-points are classified as noise.

To understand how the algorithm expands its clusters, a weaker form of reachability is required, which is defined as follows. If there is a chain of points $\{p_i\}_{i=1,...,n}$ such that $p_1 = p$, $p_n = q$ and p_{i+1} is directly reachable from p_i $\forall i$, then q is said to be density-reachable from p.

In general, the algorithm starts with a random point p. If the size of its neighbourhood is sufficient, a new cluster is started and expanded with all points that are density-reachable from p. Otherwise it is marked as noise, but it may still be made part of a cluster later if it is in the neighbourhood of a suitable point. The algorithm in full is summarized in Algorithm B.4, where we note that *RangeQuery*(p, ϵ) finds all points q such that $d(p, q) \le \epsilon$.

Algorithm B.4 **DBSCAN**

Data: D (Set of unclassified points), ϵ (Maximum radius of cluster), *minPnts* (Minimum amount of observations to form cluster)
Result: Classification of D into clusters or noise
Initialize cluster id $C = 0$;
for *each unclassified point* $p \in D$ **do**
  $N_\epsilon(p) = RangeQuery(p, \epsilon)$;
  if $\mid N_\epsilon(p) \mid \geq minPnts$ **then**
    Classify p to cluster C ;
    for *each point* $q \in N_\epsilon(p)$ **do**
      if q *is unclassified* **then**
        $N_\epsilon(q) = RangeQuery(q, \epsilon)$;
        if $\mid N_\epsilon(q) \mid \geq minPnts$ **then**
          $N_\epsilon(p) = N_\epsilon(p) \cup N_\epsilon(q)$;
      if q *is not in a cluster* **then**
        Classify q to cluster C;
    $C \leftarrow C + 1$;
  else
    Classify p as noise;

B.3. Recursive Feature Elimination

Algorithm B.4 describes an enhanced version of the RFE algorithm as used by *caret* [45]. The issue with basic RFE is that it does not account for the additional variability caused by feature selection, when it is part of the model-building process. Therefore, an extra outer loop is defined, where a resampling procedure is used to better account for this variability.

Algorithm B.5 *Recursive Feature Elimination enhanced with resampling*

for *Each Resampling Iteration* **do**
  Partition data into training and test/hold-back set via resampling;
  Tune/train the model on the training set using all predictors;
  Predict the held-back samples;
  Calculate variable importance or rankings;
  for *Each subset size* $S_i, i = 1 \ldots S$ **do**
    Keep the S_i most important variables;
    [Optional] Pre-process the data;
    Tune/train the model on the training set using S_i predictors;
    Predict the held-back samples;
    [Optional] Recalculate the rankings for each predictor
  Calculate the performance profile over the S_i using the held-back samples;
  Determine the appropriate number of predictors;
  Estimate the final list of predictors to keep in the final model;
  Fit the final model based on the optimal S_i using the original training set

B.4. Joint shrinkage property of ridge regression

A proof is given for why ridge regression simultaneously shrinks the collinear variables by an equal amount.
Suppose that we fit a ridge regression with a single variable $\boldsymbol{X}_i = \{x_1, ..., x_N\}$, outcome variable $\boldsymbol{y} = \{y_1, ..., y_N\}$ and penalty λ, which then returns a coefficient of a. From 4.38 it follows that $\left(\sum_{j=1}^{N} x_j^2 + \lambda\right)^{-1} \cdot \sum_{j=1}^{N} x_j y_j = a$.
We calculate the coefficients from 4.38 when another variable $\boldsymbol{X}_k$ is added to the ridge regression, which is identical to

the initial variable (i.e. perfect collinearity):

$$\begin{aligned}\hat{\beta}_{ridge} &= \begin{bmatrix} \sum_{j=1}^N x_j^2 + \lambda & \sum_{j=1}^N x_j^2 \\ \sum_{j=1}^N x_j^2 & \sum_{j=1}^N x_j^2 + \lambda \end{bmatrix}^{-1} \begin{bmatrix} \sum_{j=1}^N x_j y_j \\ \sum_{j=1}^N x_j y_j \end{bmatrix} \\ &\overset{*}{=} \begin{bmatrix} \frac{\sum x_j^2+\lambda}{2\lambda\sum x_j^2+\lambda^2} & -\frac{\sum x_j^2}{2\lambda\sum x_j^2+\lambda^2} \\ -\frac{\sum x_j^2}{2\lambda\sum x_j^2+\lambda^2} & \frac{\sum x_j^2+\lambda}{2\lambda\sum x_j^2+\lambda^2} \end{bmatrix} \begin{bmatrix} \sum_{j=1}^N x_j y_j \\ \sum_{j=1}^N x_j y_j \end{bmatrix},\end{aligned} \tag{B.7}$$

where after the starred equal sign we used that:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{\det \begin{bmatrix} a & b \\ c & d \end{bmatrix}} \cdot \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}. \tag{B.8}$$

Working out $\hat{\beta}_1^{ridge}$ (calculations for $\hat{\beta}_2^{ridge}$ are identical):

$$\begin{aligned}\hat{\beta}_1^{ridge} &= \sum x_j y_j \cdot \frac{\sum x_j^2 + \lambda}{2\lambda\sum x_j^2 + \lambda^2} - \sum x_j y_j \cdot \frac{\sum x_j^2}{2\lambda\sum x_j^2 + \lambda^2} \\ &= \sum x_j y_j \cdot \left(2\sum x_j^2 + \lambda\right)^{-1} \\ &= \underbrace{\sum x_j y_j \left(\sum x_j^2 + \lambda\right)^{-1}}_{=\mathrm{a}} \cdot \underbrace{\frac{\sum x_j^2 + \lambda}{2\sum x_j^2 + \lambda}}_{<1},\end{aligned} \tag{B.9}$$

so we can indeed see that every collinear variable is shrunk by the same amount. Note that the above result was evaluated for $p = 2$, but it generalizes to $p = m$ for any m.

B.5. Quantile loss minimization

Let $L_\tau(y)$ be the quantile loss function, f the PDF of Y and $\hat{\boldsymbol{y}}$ be the predictions from some model that minimizes $\mathbb{E}\left[L_\tau\left(Y-\hat{y}\right)\right]$. To solve for $\hat{y}$, first note that:

$$\mathbb{E}\left[L_\tau\left(Y-\hat{y}\right)\right] = (\tau-1)\int_{-\infty}^{\hat{y}} f(t)(t-\hat{y})dt + \tau\int_{\hat{y}}^{\infty} f(t)(t-\hat{y})dt. \tag{B.10}$$

Taking the derivative w.r.t. $\hat{y}$ and equaling to zero then gives, using the chain rule and fundamental theorem of calculus:

$$\begin{aligned}0 &= \frac{\partial}{\partial\hat{y}}\mathbb{E}\left[L_\tau\left(Y-\hat{y}\right)\right] = \tau\left(0-\int_{\hat{y}}^{\infty} f(t)dt\right) - (1-\tau)\left(0-\int_{-\infty}^{\hat{y}} f(t)dt\right) \\ &= F(\hat{y}) - \tau.\end{aligned}$$

Therefore, $\hat{y} = F^{-1}(\tau)$ is the τ-th quantile of Y.
Finally, note that a consistent estimate for the quantile may be obtained by minimizing the mean quantile loss $\frac{1}{N}\sum_{i=1}^N L_\tau(y_i - \hat{y}_i)$, since by the law of large number it converges to $\mathbb{E}\left[L_\tau\left(Y-\hat{y}\right)\right]$.

C

Appendix C

C.1. Full Data List

Name	Description
toren_outlet_temp_lucht	Tower - Outlet air temperature
koelbed_air_temp_section_1A	Fluidizer - temperature airsection 1A
koelbed_air_temp_section_1B	Fluidizer - temperature airsection 1B
koelbed_air_temp_section_1C	Fluidizer - temperature airsection 1C
koelbed_air_temp_section_1D	Fluidizer - temperature airsection 1C
koelbed_onderdruk	Fluidizer - negative pressure
koelbed_pressure_difference_1A	Fluidizer - pressure difference 1A
koelbed_pressure_transm_1A	Fluidizer - pressure transmission airsection 1A
koelbed_pressure_diff_1D	Fluidizer - pressure difference 1D
koelbed_pressure_1B	Fluidizer - pressure 1B
koelbed_pressure_diff_1C	Fluidizer - pressure difference 1C
koelbed_pressure_1C	Fluidizer - pressure 1C
koelbed_pressure_diff_1B	Fluidizer - pressure difference 1B
koelbed_level_control_valve	Fluidizer - Level Control Valve (0 = Production, 100 = CIP or emptying)
koelbed_hopper2_Ens_CaHMB	Fluidizer - Hopper/Transport system 2 - Ens CaHMB
koelbed_hopper3_Isomil	Fluidizer - Hopper/Transport system 3 - Isomil
koelbed_hopper1_Ens_Ped	Fluidizer - Hopper/Transport system 1 - Ens/Ped
toren_onderdruk	Tower - negative pressure
toren_nozzle_pressure2	Tower - Nozzle pressure 2
IND2_product_afvoer_density	Evaporator 2 - product outlet density
IND1_product_flow_op_indamper	Evaporator 1 - product flow on evaporator
toren_inlet_temp_lucht	Tower - inlet air temperature
toren_HDL1_product_flow	Tower - High Density Line 1 product flow
toren_HDL2_product_flow	Tower - High Density Line 2 product flow
toren_nozzle_pressure1	Tower - Nozzle pressure 1
IND1_steampress_TVR1_1ste_trap	Evaporator 1 - steam pressure TVR 1 (1st stairs)
IND2_steampress_TVR1_1ste_trap	Evaporator 2 - steam pressure TVR 1 (1st stairs)
toren_hoeveelheid_lucht	Tower - Amount of air in tower
IND1_product_afvoer_density	Evaporator 1 - product outlet density
toren_mixing_ratio_process_air_outlet	Tower - Mixing ratio/Humidity of process air outlet
toren_output_rate	Tower - dryer output rate
toren_H001_temp	Tower - HDL1 temperature
toren_H002_temp	Tower - HDL2 temperature
toren_gas_flow_op_brander	Tower - gas flow on burner
UHT1_flowmeter_validated	UHT 1 - Flowmeter (Validated)
UHT1_DSI_temp_regeling_validated	UHT 1 - DSI temperature regulation (Validated)
UHT1_flashvat_temp	UHT 1 - Flashvat temperature
UHT1_temp_transmitter_condensor	UHT 1 - Temperature transmitter of condensor

Name	Description
UHT1_pressure_circulatie_product_WW11	UHT 1 - circulation pressure of product (WW11)
UHT1_pressure_regeneratief_product_WW12	UHT 1 - regenerative pressure of product (WW12)
UHT1_pressure_circulation_water_WW11	UHT 1 - circulation pressure of water (WW11)
UHT1_pressure_regeneratief_water_WW12	UHT 1 - regenerative pressure of water (WW12)
UHT1_product_pressure_na_DSI	UHT 1 - product pressure after DSI
UHT1_MW_scheider_vanaf_UHT1	UHT 1 - Milk/Water separator from UHT 1
UHT1_pressure_transmitter_lobbenpomp	UHT 1 - pressure of transmitter lobe pump
UHT_T090_DMB1_weight	UHTs - Weight of T090 DMB1
UHT_T091_DMB2_weight	UHTs - Weight of T091 DMB2
UHT_inhoud_tank_T092_escape_tank	UHTs - Capacity of escape tank T092
IND1_condensaat_1	Evaporator 1 - temperature of condensate
IND1_temp_koelwater_uit_1	Evaporator 1 - outlet temperature of cooling water
IND1_steam_pressure_TVR2_1ste_trap_setpoint	Evaporator 1 - steam pressure setpoint TVR2 (1st stairs)
IND1_drukmeting_calendria1_boven	Evaporator 1 - pressure at top of calendria 1
IND1_temp_calendria1	Evaporator 1 - temperature of calendria 1
IND1_VPA_PT	Monitors pressure peaks in manifold system
IND_afvalwater_VE_12uur_avg	Evaporators - pollution units in waste water (12h average)
IND_WW001_geleidbmeter_QT002_buffertank	Evaporators - conductivity meter QT-002 of buffertank (WW001)
IND_afvalwater_VE_actueel	Evaporators - pollution units in waste water (recent)
IND1_steampressure_TVR1_2de_trap	Evaporator 1 - steam pressure TVR1 (2nd stairs)
IND1_koelwater_flow_indamper1	Evaporator 1 - flow of cooling water
IND1_temp_koelwater_in_1	Evaporator 1 - inlet temperature of cooling water
dehum_motor_actual_speed	Dehumidifier - Motor actual speed (0 - 100%)
dehum_motor_actual_speed_1	Dehumidifier - Motor actual speed (0 - 100%)
dehum_valve_actual_position	Dehumidifier - Valve actual position (0 - 100%)
dehum_humidity_actual_value	Dehumidifier - Humidity actual value
dehum_pid1_humidity_controller_process_air_Output	Dehumidifier - PID1: humidity controller process air (output)
dehum_pid1_humidity_controller_process_air_Process_value	Dehumidifier - PID1: humidity controller process air (process value)
dehum_pid2_temp_controller_reg_heater_Output	Dehumidifier - PID2: temperature controller regeneration heater (output)
dehum_pid2_temp_controller_reg_heater_Process_value	Dehumidifier - PID2: temperature controller regeneration heater (process value)
dehum_pid3_temp_controller_pre_heater_Output	Dehumidifier - PID3: temperature controller pre-heater (output)
dehum_pid3_temp_controller_pre_heater_Process_value	Dehumidifier - PID3: temperature controller pre-heater (process value)
dehum_pid4_rel_humidity_controller_pre_heater_Output	Dehumidifier - PID4: relative humidity controller pre-heater (output)
dehum_pid4_rel_humidity_controller_pre_heater_Process_value	Dehumidifier - PID4: relative humidity controller pre-heater (process value)
dehum_pid5_temp_condensate_pre_heater_Output	Dehumidifier - PID5: temperature condensate pre-heater (output)
dehum_pid5_temp_condensate_pre_heater_Process_value	Dehumidifier - PID5: temperature condensate pre-heater (process value)
dehum_mixing_ratio_inlet	Dehumidifier - Inlet Mixing ratio/Humidity
dehum_sensor_actual_value_TT141	Dehumidifier - Measured humidity at sensor TT141
dehum_sensor_actual_value_TT133	Dehumidifer - Measured humidity at sensor TT133
dehum_sensor_actual_value_TT132	Dehumidifier - Measured humidity at sensor TT132
IND2_product_flow_op_indamper	Evaporator 2 - product flow on evaporator
IND2_steampressure_TVR1_2de_trap	Evaporator 2 - steam pressure TVR1 (2nd stairs)
IND2_koelwater_flow_indamper2	Evaporator 2 - flow of cooling water
IND2_temp_koelwater_in_2	Evaporator 2 - inlet temperature of cooling water
IND2_temp_condensaat_2	Evaporator 2 - temperature of condensate
IND2_temp_koelwater_uit_2	Evaporator 2 - outlet temperature of cooling water
IND2_drukmeting_calendria1_boven	Evaporator 2 - pressure at top of calendria 1
IND2_temp_calendria1	Evaporator 2 - temperature of calendria 1
IND2_VPA_PT	Monitors pressure peaks in manifold system
UHT2_flowmeter_validated	UHT 2 - Flowmeter (Validated)
UHT2_DSI_temp_regeling_validated	UHT 2 - DSI temperature regulation (Validated)
UHT2_flashvat_temp	UHT 2 - Flashvat temperature
UHT2_temp_transmitter_condensor	UHT 2 - temperature transmitter of condensor
UHT2_pressure_circulation_product_WW11	UHT 2 - circulation pressure of product (WW11)
UHT2_pressure_regeneratief_product_WW12	UHT 2 - regenerative pressure of product (WW12)
UHT2_pressure_circulatie_water_WW11	UHT 2 - circulation pressure of water (WW11)
UHT2_pressure_regeneratief_water_WW12	UHT 2 - regenerative pressure of water (WW12)
UHT2_product_pressure_na_DSI	UHT 2 - product pressure after DSI

Name	Description
UHT2_MW_scheider_vanaf_UHT2	UHT 2 - M/W separator from UHT2
UHT2_pressure_transmitter_lobbenpomp	UHT 2 - pressure of transmitter lobe pump
toren_productflow_op_toren	Tower - productflow on tower
toren_CIP_Homo1_pressure_stap2	Tower - pressure in homo 1 step 2 (CIP)
toren_actueel_stapnummer	Tower - current stepnumber of programme
koelbed_OH003_VH003_send_to_bigbag_or_silo	Fluidizer - OH003/VH003: send product to bigbag or silo
koelbed_OH001_VH001_send_to_bigbag_or_silo	Fluidizer - OH001/VH001: send product to bigbag or silo
koelbed_OH002_VH002_send_to_bigbag_or_silo	Fluidizer - OH002/VH002: send product to bigbag or silo
koelbed_OH001_VH001_connected_to_silo_number	Fluidizer - OH001/VH001: indicates to which silo it is connected
toren_status_register_APC	Tower - registered status of APC
UHT_productie_active_UHT_actueel_stapnummer	UHTs - current stepnumber of production in active UHT
UHT1_CIP_actueel_stapnummer	UHT 1 - current step number of CIP
UHT2_CIP_actueel_stapnummer	UHT 2 - current stepnumber of CIP
toren_CIP_DMBs_and_Escapetank	Tower - DMBs and Escapetank (CIP)
toren_HDL1_production	Tower - High Density Line 1 production step number
toren_HDL2_production	Tower - High Density Line 2 production step number
UHT1_wellwater	UHT 1 - Indicates if wellwater is used for cooling
IND1_productie_indamper1_actueel_stapnummer	Evaporator 1 - current production step number
IND2_productie_indamper2_actueel_stapnummer	Evaporator 2 - current production stepnumber
UHT2_wellwater	UHT 2 - Indicates if wellwater is used for cooling
IND1_water_toevoerklep_WT011	Evaporator 1 - Indicates if water supply valve (WT011) is turned on
Hour	Current hour of the day
weekday	Current weekday
month	Current month of the year
ProductID	Current Product being produced
Production_Volume	Sum of total production volume over last period

Table C.1: Full list of production-related variables, note that perfectly collinear variables were removed beforehand.

D

Appendix D

D.1. Predictive results for top 10 products

D.1.1. OLS

OLS: Predictive point accuracy (without outliers and during active production only)																						
Type	Metrics	Overall	S564A0		0750A2		B335		B356		B385		B380		B337		B953	S254A0		P962A0		New
			Full	Spec	Full	Spec	Full	Spec	Full	Spec	Full	Spec	Full	Spec	Full	Spec	Full	Full	Spec	Full	Spec	Full
Test only	RMSE	1.2	1.15	3E26	1.07	1.75	1.25	**1.13**	0.85	0.84	1.26	1.85	1.17	1.17	1.57	1.2	1.26	1.18	**7E6**	0.88	**0.58**	1.48
	MAE	0.92	0.89	5E24	0.79	1.32	0.94	**0.85**	0.66	0.7	0.94	1.22	0.87	0.9	1.13	0.94	1	0.9	**2E5**	0.75	**0.45**	1.22
	R^2	0.17	0.14	0.4E-4	0.13	0.17	0.11	**0.07**	0.32	0.39	0.17	0.1	0.21	0.2	0.2	0.27	0.06	0.13	0.1E-4	0.35	**0.41**	0.02
All	RMSE	**0.74**	**0.62**	**1.88**	**0.83**	**0.82**	**0.79**	1E10	**0.59**	**0.63**	**0.57**	**0.62**	**0.78**	**0.41**	**0.77**	**0.59**	**1**	**0.85**	1E27	**0.53**	0.6	**1.14**
	MAE	**0.57**	**0.49**	**1.22**	**0.59**	**0.59**	**0.63**	2E8	**0.47**	**0.5**	**0.47**	**0.44**	**0.65**	**0.29**	**0.63**	**0.42**	**0.77**	**0.65**	4E25	**0.45**	0.49	**0.87**
	R^2	**0.36**	**0.44**	**0.05**	**0.21**	**0.23**	**0.28**	0.004	**0.51**	**0.48**	**0.43**	**0.34**	**0.26**	**0.55**	**0.2**	**0.46**	**0.17**	**0.24**	**1E-4**	**0.59**	0.41	**0.07**

Table D.1: Evaluation metrics for LM_{base_1} when only periods of active production are considered, where under "Type" we selected these periods among the test data only or the whole data. Furthermore, "New" denotes those products outside the top 10 that did not occur in the training data.

Type	Metrics	OLS: Predictive interval accuracy (without outliers and during active production only)																				
		Overall	S564A0		0750A2		B335		B356		B385		B380		B337		B953	S254A0		P962A0		New
			Full	Spec	Full	Spec	Full	Spec	Full	Spec	Full	Spec	Full	Spec	Full	Spec	Full	Full	Spec	Full	Spec	Full
Test only	Avg. Width	2.65	2.8	1E52	2.37	2E28	2.98	**3.11**	2.64	**1.58**	2.62	2.12	2.64	1.22	3.18	3E4	2.92	2.34	**5E4**	2	**0.76**	2.88
	Coverage	**91**	**95.9**	**81.4**	**85.2**	**87.3**	**94**	**94.7**	**93.6**	83.3	**89.1**	**72.9**	**93**	70.7	**94.7**	**82.3**	**84.6**	**91.9**	**63.2**	**83.5**	43.4	**92.1**
	Score	**3.7**	**3.06**	1E52	**4.78**	2E28	**3.48**	**3.95**	3.14	**1.65**	**3.43**	8.45	**2.8**	3.87	**3.23**	3E4	4.96	**2.56**	**5E4**	**3.15**	**0.76**	**4.32**
All	Avg. Width	**2.16**	**2.2**	**1.65**	**2.03**	**9E11**	**2.21**	7.75	**2.27**	1.65	**1.88**	**0.64**	**1.96**	**0.78**	**2.36**	**1.57**	**2.44**	**2**	2E101	**1.66**	1.46	**2.59**
	Coverage	82.8	89.4	73.9	79.8	73.6	83.6	84.6	86.9	**89.2**	73.1	48.6	77.6	**84.6**	86.2	65.4	76.7	76.9	57.9	73.5	**72.8**	74.3
	Score	4.04	3.25	**2.04**	5.4	**9E11**	4.12	8.83	**2.85**	1.75	4.92	**1.6**	4.36	**1.18**	3.5	**4.28**	**4.95**	3.63	2E101	3.66	1.57	5.19

Table D.2: Evaluation metrics of the PIs constructed from quantile regression and the LM_{base_1} model, during periods of active production.

D.1.2. LASSO

Lasso: Predictive point accuracy (without outliers and during active production only)																						
Type	Metrics	Overall	S564A0		0750A2		B335		B356		B385		B380		B337		B953	S254A0		P962A0		New
			Full	Spec	Full	Spec	Full	Spec	Full	Spec	Full	Spec	Full	Spec	Full	Spec	Full	Full	Spec	Full	Spec	Full
Test only	RMSE	1.13	1.07	2E20	1.07	1.5	1.17	**1.03**	0.79	0.61	1.2	1.5	1.03	0.95	1.46	0.75	1.16	1.09	**4E8**	0.72	**0.39**	1.33
	MAE	0.84	0.83	3E18	0.76	1.16	0.87	**0.77**	0.6	0.47	0.87	0.96	0.74	0.71	1.02	0.47	0.87	0.79	**1E7**	0.61	**0.3**	1.09
	R^2	0.22	0.17	0.4E-4	0.15	0.19	0.14	**0.09**	0.38	0.49	0.18	0.13	0.27	0.32	0.23	0.38	0.09	0.18	0.05	0.45	**0.43**	0.04
All	RMSE	**0.69**	**0.57**	**1.32**	**0.82**	**0.79**	**0.76**	2E17	**0.55**	**0.52**	**0.53**	**0.37**	**0.67**	**0.33**	**0.69**	**0.3**	**0.98**	**0.78**	2E10	**0.44**	0.44	**1.12**
	MAE	**0.52**	**0.44**	**0.87**	**0.56**	**0.53**	**0.59**	3E15	**0.43**	**0.42**	**0.43**	**0.25**	**0.55**	**0.23**	**0.54**	**0.23**	**0.71**	**0.59**	5E8	**0.37**	0.33	**0.82**
	R^2	**0.38**	**0.47**	**0.08**	**0.21**	**0.26**	**0.28**	0.004	**0.5**	**0.57**	**0.46**	**0.58**	**0.29**	**0.65**	**0.25**	**0.71**	**0.18**	**0.29**	**0.08**	**0.62**	0.42	**0.08**

Table D.3: Evaluation metrics for LASSO when only periods of active production are considered.

Type	Metrics	Lasso: Predictive interval accuracy (without outliers and during active production only)																			
		Overall	S564A0		0750A2		B335		B356		B385		B380		B337		B953	S254A0		P962A0		New
			Full	Spec	Full	Spec	Full	Spec	Full	Spec	Full	Spec	Full	Spec	Full	Spec	Full	Full	Spec	Full	Spec	Full
Test only	Avg. Width	**5.61**	5.02	4E20	**7**	Inf	6.72	**4.93**	4.61	3.18	5.28	14.3	5.6	5.08	6.1	2E17	5.09	**3.9**	**5E36**	4.15	1.43	4.9
	Coverage	**96.7**	**97.7**	**85.1**	**95.1**	**97.4**	**98.7**	**94.1**	96.5	89.3	**98.2**	**90.9**	**99.2**	**95**	**99.2**	**83.5**	**91.5**	91	63.8	97.1	**80.5**	**88.5**
	Score	**6.28**	5.14	1E22	**8.26**	Inf	7.1	**6.06**	4.99	4.44	5.64	20.2	5.63	5.31	6.21	2E17	8.03	**6.12**	**1E39**	4.15	2.1	**7.42**
All	Avg. Width	1E12	**3.39**	**3.7**	6E12	**3E116**	**3.85**	1E55	**3.3**	**2.34**	**3**	**0.73**	**3.28**	**1.09**	**5.75**	**1E7**	**3.89**	4E11	1E93	**3.45**	**0.89**	**4.18**
	Coverage	93.8	97	74.5	86.7	88.2	95.8	93	**96.7**	**89.3**	92.1	68.6	94.4	82.9	85.3	77.9	90.4	**95.4**	**67.1**	**97.2**	60.4	83.9
	Score	1E12	**3.81**	**15.6**	6E12	**3E116**	**4.92**	1E55	**3.9**	**2.8**	**4.3**	**2.15**	**4.18**	**2.21**	**6.02**	**6E9**	**6.6**	4E11	3E95	**3.79**	**1.41**	7.58

Table D.4: Evaluation metrics of the PIs constructed from split conformal inference (averaged over 10 runs) and the LASSO model, during periods of active production.

D.1.3. MARS

MARS: Predictive point accuracy (Active Production)												
Metrics	Overall	S564A0	0750A2	B335	B356	B385	B380	B337	B953	S254A0	P962A0	New
RMSE	0.57	0.59	0.84	0.55	0.51	0.32	0.28	0.33	0.99	0.4	0.37	1.19
MAE	0.32	0.37	0.43	0.3	0.33	0.21	0.2	0.23	0.53	0.27	0.26	0.7
R^2	0.43	0.37	0.21	0.43	0.38	0.67	0.72	0.62	0.19	0.6	0.5	0.09

Table D.5: Point accuracy for the MARS model during active production. Furthermore, we limited both training and test data to active production periods, since this gave the best results.

MARS: Predictive interval accuracy (Active Production)											
Metrics	Overall	S564A0	0750A2	B335	B356	B385	B380	B337	B953	S254A0	P962A0	New
Width	3.64	4.48	3.85	4.41	3.25	2.55	2.69	4.18	3.52	3.03	2.08	3.9
Coverage	93.8	97.7	91.2	96.6	93.8	90.8	94.7	99.2	84.6	93.3	89.3	85.4
Score	4.57	4.81	6.27	5.02	3.39	3.31	2.86	4.18	7.85	3.14	2.57	9.63

Table D.6: PI accuracy during active production periods, using S-CI for MARS (the results were averaged over 10 runs). Furthermore, we limited only the test data to active production periods, since this gave the best results.

D.1.4. Random Forest

RF: Predictive point accuracy (Active Production)												
Metrics	Overall	S564A0	0750A2	B335	B356	B385	B380	B337	B953	S254A0	P962A0	New
RMSE	0.53	0.39	0.81	0.57	0.32	0.3	0.29	0.34	0.96	0.44	0.49	1.07
MAE	0.28	0.23	0.41	0.29	0.25	0.18	0.18	0.25	0.46	0.32	0.3	0.61
R^2	0.5	0.58	0.2	0.39	0.77	0.71	0.72	0.69	0.23	0.58	0.45	0.09

Table D.7: Overall point accuracy for the RF model. Furthermore, we limited both training and test data to active production periods, since this gave the best results.

RF: Predictive interval accuracy (Active Production)													
Method	Metrics	Overall	S564A0	0750A2	B335	B356	B385	B380	B337	B953	S254A0	P962A0	New
S-CI	Width	**0.98**	**1.02**	**0.96**	**1.04**	**0.95**	**0.89**	**0.92**	**1.12**	**0.94**	**1**	**0.81**	**1.31**
	Coverage	88.1	92.5	81.7	89	86.4	93.3	95	92.4	80.3	84.3	79.2	77.6
	Score	**3.32**	**1.8**	8.17	**3.75**	**0.96**	**1.67**	**0.92**	**1.13**	10.5	**1**	**0.81**	13.1
QRF	Width	2.32	2.63	1.86	2.45	2.56	2	2.07	2.77	2	2.54	2.21	2.97
	Coverage	**96.6**	**99**	**90.2**	**95.9**	**99.2**	**98.4**	**99.4**	**99.7**	**88.6**	**99.8**	**99.8**	**88.1**
	Score	4.04	2.67	**8.05**	3.98	2.56	2.46	2.07	2.77	**9.12**	2.54	2.21	**12.5**

Table D.8: PI accuracy during active production periods for RF model, using the S-CI and QRF approaches. Furthermore, we limited only the test data to active production periods, since this gave the best results.

D.1.5. Cluster-based approach

Cluster-approach: Predictive point/interval accuracy (Active Production)													
Type	Metrics	Overall	S565A0	0750A2	B335	B356	B385	B380	B337	B953	S254A0	P962A0	New
Point	RMSE	0.25	0.25	0.27	0.25	0.26	0.27	0.21	0.23	0.31	0.2	0.17	0.29
	MAE	0.18	0.19	0.18	0.17	0.19	0.16	0.14	0.17	0.21	0.15	0.12	0.2
	R^2	0.89	0.83	0.93	0.88	0.8	0.77	0.84	0.8	0.95	0.85	0.88	0.95
Interval	Cvrg	99.3	99.4	98.9	99.2	99.2	99.7	99.3	99.4	98.6	99.8	99.6	100
	Width	1.59	1.67	1.55	1.54	1.77	1.39	1.32	1.45	1.83	1.47	1.33	2.7
	Score	1.59	1.67	1.57	1.54	1.77	1.4	1.32	1.45	1.83	1.47	1.33	2.7

Table D.9: Predictive performance of the cluster-based approach with RF model for periods of active production, using the best clustering configuration from the overall scenario.

Bibliography

[1] Energy savings toolbox – an energy audit manual and tool. *CIPEC*.

[2] International performance measurement and verification protocol: Concepts and options for determining energy and water savings. *EVO*, 2012.

[3] Energy management systems — measuring energy performance using energy baselines (enb) and energy performance indicators (enpi) — general principles and guidance. *ISO*, 2014.

[4] Energy management system (enms) guidebook for local authorities. *C4S*, 2018.

[5] Certificatieschema energiemanagementsystemen volgens iso 50001:2018. *SCCM*, 2019.

[6] R.E. Abdel-Aal. Univariate modeling and forecasting of monthly energy demand time series using abductive and neural networks. *Computers Industrial Engineering*, 54, 2008. ISSN 03608352. doi: 10.1016/j.cie.2007.10.020.

[7] Mortaza Aghbashlo, Soleiman Hosseinpour, and Arun S. Mujumdar. Application of artificial neural networks (anns) in drying technology: A comprehensive review. *Drying Technology*, 33, 2015. ISSN 0737-3937. doi: 10.1080/07373937.2015.1036288.

[8] Muhammad Waseem Ahmad, Monjur Mourshed, and Yacine Rezgui. Trees vs neurons: Comparison between random forest and ann for high-resolution prediction of building energy consumption. *Energy and Buildings*, 147, 2017. ISSN 03787788. doi: 10.1016/j.enbuild.2017.04.038.

[9] N. Amral, C. S. Ozveren, and D. King. Short term load forecasting using multiple linear regression. IEEE, 2007. ISBN 978-1-905593-36-1. doi: 10.1109/UPEC.2007.4469121.

[10] Fabrizio Ascione, Nicola Bianco, Claudio De Stasio, Gerardo Maria Mauro, and Giuseppe Peter Vanoli. Artificial neural networks to predict energy performance and retrofit scenarios for any member of a building category: A novel approach. *Energy*, 118, 2017. ISSN 03605442. doi: 10.1016/j.energy.2016.10.126.

[11] A. Azadeh, S.F. Ghaderi, and S. Sohrabkhani. Annual electricity consumption forecasting by neural network in high energy consuming industrial sectors. *Energy Conversion and Management*, 49, 2008. ISSN 01968904. doi: 10.1016/j.enconman.2008.01.035.

[12] Anshul Bansal, Susheel Kaushik Rompikuntla, Jaganadh Gopinadhan, Amanpreet Kaur, and Zahoor Ahamed Kazi. Energy consumption forecasting for smart meters. *Cognizant Technology Solutions*, 2015.

[13] Mohsen Beigi, Mehdi Torki-Harchegani, and Mojtaba Tohidi. Experimental and ann modeling investigations of energy traits for rough rice drying. *Energy*, 141, 2017. ISSN 03605442. doi: 10.1016/j.energy.2017.12.004.

[14] Irina Boiarkina, Nick Depree, Arrian Prince-Pike, Wei Yu, David I. Wilson, and Brent R. Young. Using big data in industrial milk powder process systems, 2018.

[15] George E.P. Box, Gwilym M. Jenkins, Gregory C. Reinsel, and Greta M. Ljung. *Time Series Analysis: Forecasting and Control*. Wiley, 5th edition, 2016.

[16] Silvia Bravo and Angel H. Moreno. A random forest approach for predicting the microwave drying process of amaranth seeds. *IEEE 2nd International Conference on Information and Computer Technologies*, pages 25–29, 2019.

[17] Tiberiu Catalina, Vlad Iordache, and Bogdan Caracaleanu. Multiple regression model for fast prediction of the heating energy demand. *Energy and Buildings*, 57, 2013. ISSN 03787788. doi: 10.1016/j.enbuild.2012.11.010.

[18] A.E. Clements, A.S. Hurn, and Z. Li. Forecasting day-ahead electricity load using a multiple equation time series approach. *European Journal of Operational Research*, 251, 2016. ISSN 03772217. doi: 10.1016/j.ejor.2015.12.030.

[19] Hosain Darvishi. Energy consumption and mathematical modeling of microwave drying of potato slices. *Agric Eng Int: CIGR Journal*, 14, 2012.

[20] Sijmen de Jong. Simpls: An alternative approach to partial least squares regression. *Chemometrics and Intelligent Laboratory Systems*, 18:251–263, 1993. ISSN 01697439. doi: 10.1016/0169-7439(93)85002-X.

[21] Chirag Deb, Fan Zhang, Junjing Yang, Siew Eang Lee, and Kwok Wei Shah. A review on time series forecasting techniques for building energy consumption. *Renewable and Sustainable Energy Reviews*, 74, 2017. ISSN 13640321. doi: 10.1016/j.rser.2017.02.085.

[22] Ravinesh Deo, Pijush Samui, and Sanjiban Sekhar Roy. *Predictive Modelling for Energy Management and Power Systems Engineering*. Elsevier, 1st edition, 2020.

[23] L.L. Doove, S. Van Buuren, and E. Dusseldorp. Recursive partitioning for missing data imputation in the presence of interaction effects. *Computational Statistics Data Analysis*, 72, 2014. ISSN 01679473. doi: 10.1016/j.csda.2013.10.025.

[24] J. C. Dunn. Well-separated clusters and optimal fuzzy partitions. *Journal of Cybernetics*, 4, 1974. ISSN 0022-0280. doi: 10.1080/01969727408546059.

[25] Betul Bektas Ekici and U. Teoman Aksoy. Prediction of building energy consumption by using artificial neural networks. *Advances in Engineering Software*, 40, 2009. ISSN 09659978. doi: 10.1016/j.advengsoft.2008.05.003.

[26] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. *A density-based algorithm for discovering clusters in large spatial databases with noise*. AAAI Press, 1996.

[27] Cheng Fan, Fu Xiao, and Shengwei Wang. Development of prediction models for next-day building energy consumption and peak power demand using data mining techniques. *Applied Energy*, 127, 2014. ISSN 03062619. doi: 10.1016/j.apenergy.2014.04.016.

[28] Cheng Fan, Meiling Chen, Xinghua Wang, Jiayuan Wang, and Bufu Huang. A review on data preprocessing techniques toward efficient and reliable knowledge discovery from building operational data. *Frontiers in Energy Research*, 9, 2021. ISSN 2296-598X. doi: 10.3389/fenrg.2021.652801.

[29] Aurélie Foucquier, Sylvain Robert, Frédéric Suard, Louis Stéphan, and Arnaud Jay. State of the art in building modelling and energy performances prediction: A review. *Renewable and Sustainable Energy Reviews*, 23, 2013. ISSN 13640321. doi: 10.1016/j.rser.2013.03.004.

[30] John Fox and Georges Monette. Generalized collinearity diagnostics. *Journal of the American Statistical Association*, 87:178–183, 1992. ISSN 01621459. doi: 10.2307/2290467.

[31] Jerome Friedman, Trevor Hastie, Rob Tibshirani, Balasubramanian Narasimhan, Kenneth Tay, Noah Simon, and Junyang Qian. Package 'glmnet', 2021.

[32] Iman Ghalehkhondabi, Ehsan Ardjmand, Gary R. Weckman, and William A. Young. An overview of energy demand forecasting methods published in 2005–2015. *Energy Systems*, 8, 2017. ISSN 1868-3967. doi: 10.1007/s12667-016-0203-y.

[33] E. Giacone and S. Mancò. Energy efficiency measurement in industrial processes. *Energy*, 38, 2012. ISSN 03605442. doi: 10.1016/j.energy.2011.11.054.

[34] Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102, 2007. ISSN 0162-1459. doi: 10.1198/016214506000001437.

[35] Michael Hahsler. Package "dbscan", 2022.

[36] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer New York, 2009. ISBN 978-0-387-84857-0. doi: 10.1007/978-0-387-84858-7.

[37] Rob J. Hyndman and Anne B. Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22, 2006. ISSN 01692070. doi: 10.1016/j.ijforecast.2006.03.001.

[38] Z. Ismail, F. Jamaluddin, and F. Jamaludin. Time series regression model for forecasting malaysian electricity load demand. *Asian Journal of Mathematics and Statistics*, 1:139–149, 2008.

[39] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning with Applications in R*, volume 103. Springer New York, 2013. ISBN 978-1-4614-7137-0. doi: 10.1007/978-1-4614-7138-7.

[40] Md. Imran H. Khan, Shyam S. Sablani, M. U. H. Joardder, and M. A. Karim. Application of machine learning-based approach in food drying: opportunities and challenges. *Drying Technology*, 2020. ISSN 0737-3937. doi: 10.1080/07373937.2020.1853152.

[41] Roger Koenker. *Quantile Regression*. Cambridge University Press, 2005. ISBN 9780511754098. doi: 10.1017/CBO9780511754098.

[42] Roger Koenker. Package 'quantreg', 2021.

[43] Irena Koprinska, Mashud Rana, and Vassilios G. Agelidis. Correlation and instance based feature selection for electricity load forecasting. *Knowledge-Based Systems*, 82, 2015. ISSN 09507051. doi: 10.1016/j.knosys.2015.02.017.

[44] William H. Kruskal and W. Allen Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, 47, 1952. ISSN 0162-1459. doi: 10.1080/01621459.1952.10483441.

[45] Max Kuhn. Package 'caret', 2021.

[46] Sricharan Kumar and Ashok Srivastava. Bootstrap prediction intervals in non-parametric regression with applications to anomaly detection. NASA, 2012.

[47] Jing Lei, Max G'Sell, Alessandro Rinaldo, Ryan J. Tibshirani, and Larry Wasserman. Distribution-free predictive inference for regression. *Journal of the American Statistical Association*, 113, 2018. ISSN 0162-1459. doi: 10.1080/01621459.2017.1307116.

[48] Qiong Li and Qinglin Meng. Development and application of hourly building cooling load prediction model. IEEE, 2010. ISBN 978-1-4244-7831-6. doi: 10.1109/ICAEE.2010.5557536.

[49] Yanchi Liu, Zhongmou Li, Hui Xiong, Xuedong Gao, and Junjie Wu. Understanding of internal clustering validation measures. pages 911–916. IEEE, 2010. ISBN 978-1-4244-9131-5. doi: 10.1109/ICDM.2010.35.

[50] Zi-Liang Liu, Jun-Wen Bai, Shu-Xi Wang, Jian-Sheng Meng, Hui Wang, Xian-Long Yu, Zhen-Jiang Gao, and Hong-Wei Xiao. Prediction of energy and exergy of mushroom slices drying in hot air impingement dryer by artificial neural network. *Drying Technology*, 38, 2020. ISSN 0737-3937. doi: 10.1080/07373937.2019.1607873.

[51] Francisco Martínez-Álvarez, Alicia Troncoso, Gualberto Asencio-Cortés, and José Riquelme. A survey on data mining techniques applied to electricity-related time series forecasting. *Energies*, 8, 2015. ISSN 1996-1073. doi: 10.3390/en81112361.

[52] Alex Martynenko and N. N. Misra. Machine learning in drying. *Drying Technology*, 38, 2020. ISSN 0737-3937. doi: 10.1080/07373937.2019.1690502.

[53] Gökan May, Ilaria Barletta, Bojan Stahl, and Marco Taisch. Energy management in production: A novel method to develop key performance indicators for improving energy efficiency. *Applied Energy*, 149, 2015. ISSN 03062619. doi: 10.1016/j.apenergy.2015.03.065.

[54] Nicolai Meinshausen. Quantile regression forests. *Journal of Machine Learning Research*, 7:983–999, 2006.

[55] Ming Meng, Dongxiao Niu, and Wei Sun. Forecasting monthly electric energy consumption using feature extraction. *Energies*, 4, 2011. ISSN 1996-1073. doi: 10.3390/en4101495.

[56] Norizan Mohamed, Maizah Hura Ahmad, Suhartono, and Wan Muhammad Amir Wan Ahmad. Forecasting short term load demand using multilayer feed-forward (mlff) neural network model. *Applied Mathematical Sciences*, 6:5359–5368, 2012.

[57] Amir Mosavi and Abdullah Bahmani. Energy consumption prediction using machine learning; a review, 2019.

[58] Tayyeb Nazghelichi, Mortaza Aghbashlo, Mohammad Hossein Kianmehr, and Mahmoud Omid. Prediction of energy and exergy of carrot cubes in a fluidized bed dryer by artificial neural networks. *Drying Technology*, 29, 2011. ISSN 0737-3937. doi: 10.1080/07373937.2010.494237.

[59] Noelia Oses, Aritz Legarretaetxebarria, Marco Quartulli, Igor García, and Mikel Serrano. Uncertainty reduction in measuring and verification of energy savings by statistical learning in manufacturing environments. *International Journal on Interactive Design and Manufacturing (IJIDeM)*, 10, 2016. ISSN 1955-2513. doi: 10.1007/s12008-016-0302-y.

[60] R. Perez-Chacon, R. L. Talavera-Llames, F. Martinez-Alvarez, and A. Troncoso. Finding electric energy consumption patterns in big time series data, 2016.

[61] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20, 1987. ISSN 03770427. doi: 10.1016/0377-0427(87)90125-7.

[62] Jörg Sander, Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu. Density-based clustering in spatial databases: The algorithm gdbscan and its applications. *Data Mining and Knowledge Discovery*, 2:169–194, 1998. ISSN 13845810. doi: 10.1023/A:1009745219419.

[63] Alexis Sardá-Espinosa. Comparing time-series clustering algorithms in r using the dtwclust package, 2019.

[64] Julio R. Gómez Sarduy, Katia Gregio Di Santo, and Marco Antonio Saidel. Linear and non-linear methods for prediction of peak load at university of são paulo. *Measurement*, 78, 1 2016. ISSN 02632241. doi: 10.1016/j.measurement.2015.09.053.

[65] John E. Seem. Pattern recognition algorithm for determining days of the week with similar energy consumption profiles. *Energy and Buildings*, 37, 2005. ISSN 03787788. doi: 10.1016/j.enbuild.2004.04.004.

[66] John E. Seem. Using intelligent data analysis to detect abnormal energy consumption in buildings. *Energy and Buildings*, 39, 2007. ISSN 03787788. doi: 10.1016/j.enbuild.2006.03.033.

[67] Mel Keytingan M. Shapi, Nor Azuana Ramli, and Lilik J. Awalin. Energy consumption prediction by using machine learning for smart building: Case study in malaysia. *Developments in the Built Environment*, 5:–, 2021. ISSN 26661659. doi: 10.1016/j.dibe.2020.100037.

[68] Maya Shelke and Prashant D. Thakare. Short term load forecasting by using data mining techniques. *International Journal of Science and Research*, 3:1363–1367, 2014.

[69] Robert H. Shumway and David S. Stoffer. *Time Series Analysis and Its Applications with R Examples*. Springer International Publishing, 4th edition, 2017. ISBN 978-3-319-52451-1. doi: 10.1007/978-3-319-52452-8.

[70] Shailendra Singh and Abdulsalam Yassine. Big data mining of energy time series for behavioral analytics and energy consumption forecasting. *Energies*, 11, 2018. ISSN 1996-1073. doi: 10.3390/en11020452.

[71] Aldo Solari and Vera Djordjilovic. Multi split conformal prediction. 2021.

[72] Michael C. Thomas, Wenbo Zhu, and Jose A. Romagnoli. Data mining and clustering in chemical process databases for monitoring and knowledge discovery. *Journal of Process Control*, 67, 2018. ISSN 09591524. doi: 10.1016/j.jprocont.2017.02.006.

[73] Samir Touzani, Baptiste Ravache, Eliot Crowe, and Jessica Granderson. Statistical change detection of building energy consumption: Applications to savings estimation. *Energy and Buildings*, 185, 2019. ISSN 03787788. doi: 10.1016/j.enbuild.2018.12.020.

[74] Hristos Tyralis, Georgios Karakatsanis, Katerina Tzouka, and Nikos Mamassis. Exploratory data analysis of the electrical energy demand in the time domain in greece. *Energy*, 134, 2017. ISSN 03605442. doi: 10.1016/j.energy.2017.06.074.

[75] Stef van Buuren and Karin Groothuis-Oudshoorn. mice : Multivariate imputation by chained equations in r. *Journal of Statistical Software*, 45, 2011. ISSN 1548-7660. doi: 10.18637/jss.v045.i03.

[76] Jaeyeong Yoo and Kyeon Hur. Load forecast model switching scheme for improved robustness to changes in building energy consumption patterns. *Energies*, 6, 2013. ISSN 1996-1073. doi: 10.3390/en6031329.

[77] Zhun Yu, Fariborz Haghighat, Benjamin C.M. Fung, and Hiroshi Yoshino. A decision tree method for building energy demand modeling. *Energy and Buildings*, 42, 2010. ISSN 03787788. doi: 10.1016/j.enbuild.2010.04.006.

[78] Chuan Zhang, Liwei Cao, and Alessandro Romagnoli. On the feature engineering of building energy data mining. *Sustainable Cities and Society*, 39, 2018. ISSN 22106707. doi: 10.1016/j.scs.2018.02.016.

[79] Pei Zhang, Xiaoyu Wu, Xiaojun Wang, and Sheng Bi. Short-term load forecasting based on big data technologies. *CSEE Journal of Power and Energy Systems*, 1, 2015. ISSN 2096-0042. doi: 10.17775/CSEEJPES.2015.00036.

[80] Jinlin Zhu, Zhiqiang Ge, Zhihuan Song, and Furong Gao. Review and big data perspectives on robust data mining approaches for industrial process modeling with outliers and missing data. *Annual Reviews in Control*, 46, 2018. ISSN 13675788. doi: 10.1016/j.arcontrol.2018.09.003.