VRIJE UNIVERSITEIT AMSTERDAM

MASTER PROJECT
MSC BUSINESS ANALYTICS

Production Duration Estimation in a Mass-Customization Age

A Machine Learning Approach

Author:
D.J.M. VAN WEERDENBURG

Graduation supervisor:
Dr. J. BERKHOUT

Second reader:
Prof. Dr. R. VAN DER MEI

External supervisor:
Ir. J. STOLZE MSc

September 30, 2019

Production Duration Estimation in a Mass-Customization Age

D.J.M. VAN WEERDENBURG

Master project Business Analytics

Vrije Universiteit Amsterdam
Faculty of Science
De Boelelaan 1085
1081HV Amsterdam

ENGIE Services
Industrial Automation
Albert Heijnweg 1
1507EH Zaandam

VRIJE UNIVERSITEIT AMSTERDAM

Abstract

Production Duration Estimation in a Mass-Customization Age

by D.J.M. VAN WEERDENBURG

In today's animal feed industry, the schedulers deal with a lot of complexity. Some products need to be produced at certain production lines, some products cannot be produced in a row due to contamination issues and orders need to be delivered in time. Therefore, there is a demand for a scheduling algorithm that takes these complexities into account and minimizes the total manufacturing time.

Experts at ENGIE Industrial Automation develop such a scheduling algorithm, for which accurate estimations of the production durations are needed. The current estimations are created based on the duration of the previous batches of the same produced article. A batch is a quantity of product that is produced in a single production step. However, customers are increasingly demanding tailored products, a trend known as mass-customization [1], which results in too few produced batches of the same article. Therefore, there are often too few data points available for an accurate estimation of the current approach. Besides, the current approach is not able to give an estimation for new products. This is the direct motivation for this study.

Since products are specified by their nutritions instead of ingredients, the ingredient composition of the same article could change over time, which could result in variation of the production durations. Besides, changes in batch-related machine settings (setpoints), the weather and the quality of the harvest can potentially affect the production duration. Therefore, the research question of this research is formulated as follows:

> *Is it possible to enhance production duration estimations by incorporating the ingredient composition of products, seasonal effects, and machine settings?*

The results of this research show that it is possible to enhance the production duration estimations. In all production steps, the duration could be estimated significantly better than the benchmark. Except for the duration estimation of the transportation system, which is as good as the benchmark.

The estimations were divided into complex machine learning models (for production durations) and simple time-based models (for transportation durations). For the complex machine learning models, we can conclude that all models estimate the durations for all production steps significantly better than the benchmark, except for the Neural Network, which estimates the pressing duration at press line 3 worse than the benchmark.

Extra Trees and Robust Regression show generally good results when less than one year of training data is available. Also, the Robust Regression model performs generally well in cases where Extra Trees is less accurate. Therefore, ENGIE Industrial Automation is advised to implement these two models and to use them for the production duration for which they are appropriate according to their performance described in the results of this research.

Finally, we can conclude that the models proposed in this research can help the scheduling algorithm of ENGIE Industrial Automation to be more accurate.

Acknowledgements

This master project is part of the curriculum of the master Business Analytics at the Vrije Universiteit Amsterdam. Students are supposed to complete a six-month internship at a company. During my internship at ENGIE Industrial Automation, I had the opportunity to work on a concrete problem in the industry.

This research is established by supervision of Joost Berkhout on behalf of the Vrije Universiteit and Jan Stolze on behalf of ENGIE Industrial Automation. This research is part of a scheduling project in which the resources of an industrial plant in the animal feed industry are used as efficiently as possible to deliver all orders in time. Besides, this scheduling algorithm helps the schedulers of the plant to coop with the daily scheduling complexity due to, for example, contamination rules and production line restrictions.

I would like to thank Joost Berkhout for his ideas and guidance throughout the research. He helped me with ideas and gave me concrete feedback on this paper, which helped me a lot. Also, I would like to thank Rob van der Mei, who is the second reader on behalf of the Vrije Universiteit, for his involvement in this project.

Furthermore, I would like to thank Jan Stolze, who is a software engineer and mathematician at ENGIE Industrial Automation, for his time and effort to make me understand the MES Toolbox (software application that runs the plant) and to provide me information about the particular animal feed plant. Since he will implement the Machine Learning models in the MES Toolbox, he kept in mind that my approach should be able to coop with certain effects that are likely to occur while my models are running in the plant in the future. For example, the window-based outlier filtering technique was invented in collaboration. I think that understanding the effects of certain modelling choices helps to improve the usability of the models in the future. Jan helped me to bring this project to the next level.

In addition, I would like to thank Siem Broersen and René Kruijer for their enthusiasm and extensive explanations of the operation of the animal feed plant. Besides them, Viktor Klein also showed confidence in me, the project and data science in general.

Finally, I am happy with the possibility I got to show my results in the pilot plant. I would like to thank them for their tour in the plant. It was very nice to see the real machines: "the data came to life".

I hope you like this project as much as I do.

Sascha van Weerdenburg
Zaandam, September 2019.

Contents

List of Abbreviations

GML	**G**rinder-**M**ixer **L**ine
PL	**P**ress **L**ine
PO	**P**roduction **O**rder
CO	**C**ustomer **O**rder
PFC	**P**rocedural **F**unction **C**harts
BOA	Certain type of pre-compactor
ML	**M**achine **L**earning
CART	**C**lassfication **A**nd **R**egression **T**rees
GBM	**G**radient **B**oosting **M**achine
XGBoost	e**X**treme **G**radient **Boost**ing
ANN	**A**rtficial **N**eural **N**etwork
MLP	**M**ulti-**L**ayer **P**erceptron
ReLU	**Re**ctified **L**inear **U**nit
GD	**G**radient **D**escent
SGD	**S**tochastic **G**radient **D**escent
AdaGrad	**Ada**ptive **G**radient **A**lgorithm
RMSProp	**R**oot **M**ean **S**quare **P**ropagation
Adam	Adaptive moment estimation
GOSS	**G**radient-based **O**ne-**S**ide **S**ampling
EFB	**E**xclusive **F**eature **B**undling
MAE	**M**ean **A**bsolute **E**rror
MSE	**M**ean **S**quare **E**rror
MAPE	**M**ean **A**bsolute **P**ercentage **E**rror

1 Introduction

1.1 Business context and problem statement

ENGIE is a global energy company, a leading provider of electricity, natural gas, and energy services. The department Industrial Automation focusses on designing, realizing, maintaining and exploiting software solutions for the automation of industrial plants. By automating the production process, the total production time is significantly reduced. To optimize the production process even more, a scheduling algorithm is created to allocate machines as efficient as possible. This algorithm assumes known production durations for each batch (quantity of product) on each machine. ENGIE Industrial Automation is currently testing this algorithm in an industrial plant in the animal feed industry. Hereby, the production durations are estimated by taking the moving median over the same product type. However, this approach is unsatisfactory, because customers are increasingly demanding tailored products. In literature, this trend is known as mass-customization [1]. As a result, there are often too few data points available per product type, or even none in case of new products.

A lot of research has been done on scheduling algorithms in general. Most of these algorithms assume deterministic production durations, others assume a certain distribution to hold which should comprehend the uncertainty in the durations. However, little research is done on the usage of the ingredient composition, seasonal effects and machine settings for the estimation of the production durations in industrial plants. Therefore, it would be interesting to investigate if it is possible to enhance production duration estimations by incorporating these features.

Besides the effect of mass-customization, there are more reasons for ENGIE Industrial Automation to be interested in a model to estimate production durations based on the ingredient composition, seasonal effects, and machine settings. For example, the ingredient composition of existing product types can deviate over time. A product is namely defined by its nutrients and therefore the ingredient composition can change if some ingredients are cheaper than others. This suggests that it would be better to use the ingredient composition of products instead of taking the moving median over the product type. Moreover, the quality and texture of organic ingredients can deviate depending on, for example, the amount of water and sunlight it got during its growth. But also, the temperature and the humidity in the industrial plant affect the production process. In addition, it is not uncommon to change machine settings for the production of a certain product. This results in a lot of possible features that are likely to affect production durations. Not only the final model created in this research would be relevant for ENGIE Industrial Automation, but also the analysis of which features most affect the production durations and how.

The production durations are already estimated by ENGIE Industrial Automation for scheduling purposes. However, the current method is relatively simple and does not use all of the above-mentioned variables. Therefore, it could likely be improved. Besides, the current method is not able to provide an estimation for newly produced articles. The current estimation method will be used as a benchmark for the estimations proposed in this research.

1.2 Research question

The goal of this research is to estimate production durations by using features that are available for all products, including newly created products. This results in the following research question:

> *Is it possible to enhance production duration estimations by incorporating the ingredient composition of products, seasonal effects, and machine settings?*

By providing good estimations, the uncertainty in the scheduling algorithm will be reduced, which enables the algorithm to plan more accurately. In this way, the planning algorithm could perform better and this helps to reach the final goal: deliver all orders in time and decrease the total manufacturing time.

In addition, the analysis of the importance of features would be very relevant for ENGIE Industrial Automation as well, because it could confirm the suggested influence of abovementioned variables and it provides insights in the degree of their influence. This results in the following two sub-questions:

1. *What are the explanatory variables for the production duration?*
2. *Which models lead to better production duration predictions compared to the benchmark approach?*

Since the production durations will be used in the scheduling algorithm, the predicted durations should be defined in the same way as assumed by the scheduling algorithm, which is clearly described by the current production duration estimations. For these predictions, only features can be used which are available at the moment of scheduling, and therefore, the model can only use variables which are known before the moment of execution of the corresponding production step.

This project will focus on a single industrial plant in the animal feed industry. However, the results can most likely be generalized to other industrial plants in the animal feed industry, which means that ENGIE Industrial Automation could reuse the results for other clients.

1.3 Research approach

As mentioned above, it is assumed that there exist variables, like the ingredients, seasonal effects, and machine settings, that influence the durations of animal-feed production steps. However, the exact relationship is not defined. Since there is a lot of data available, the suggested relationship could most likely be "learned" from past production durations. This approach is called *machine learning*, see the definition given by T. Mitchell (1997) below:

> "A computer program is said to **learn** from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." [2]

To be able to apply machine learning for the estimation of production durations, seven steps are taken in this research, see Figure 1.1.

The first step is *problem understanding*, of which an introduction was given in this section. In the next section, an introduction is given to concepts in industrial automation and the production process of the specific animal-feed industrial plant. In addition, an overview

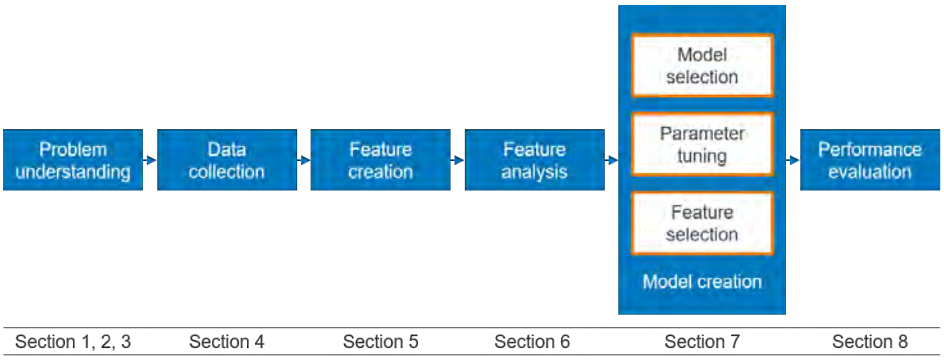


FIGURE 1.1: Visualization of the research approach with references to the related sections in this report.

of the relevant literature is given in Section 3. When we obtained the required background knowledge of the production process in the animal feed industry, we can continue with the second step in the research approach, *data collection*. This process and the collected data are described in Section 4.

The data variables that are used in a machine learning model are commonly called *features*. These features can be similar to the original data attributes, but can also be created by, for example, aggregations and/or subtractions. This process is the third step of the research approach and will be described in Section 5.

The fourth step is *feature analysis*. In this step, the created features are analysed. For example, how the features are correlated to each other and the production durations. The feature analysis is described in Section 6.

The next step in the research approach is the *model creation* step. Some state-of-the-art machine learning models that are relevant for this research are described in Section 7. In addition, the experimental setup is discussed in that section too.

The *performance evaluation* takes place in Section 8. The results of different hyperparameter settings and feature sets for different kind of models are evaluated, by which the best hyperparameters and features will be selected for each model. It is also the section in which we will answer the first sub-question of this research. In addition, the performance on unseen data will be examined by performing the final models on a test set of samples that were left out so far. For a new product, the current approach for estimating production durations, the benchmark, is not able to provide an estimation. Therefore, the test samples will be split into two sets: batches with benchmark estimation and batches without benchmark estimation. The performance of the final models will be compared with the performance of the benchmark on batches with benchmark estimation. On top of that, the results will be compared to the results on the batches without benchmark estimation to compare the performance of the models on new products. Based on these two performance measures, the best model can be determined, which provides the answer to the second sub-question and the research question.

Finally, the discussion and the conclusion are given in Section 9 and Section 10, respectively.

2 Preliminaries

The production durations will be estimated for a single industrial plant in the animal feed industry, which is further denoted by plant X. Plant X will serve as an example for other industrial plants. To understand how the plant and its production process looks like, the manufacturing process of plant X will be described in this section.

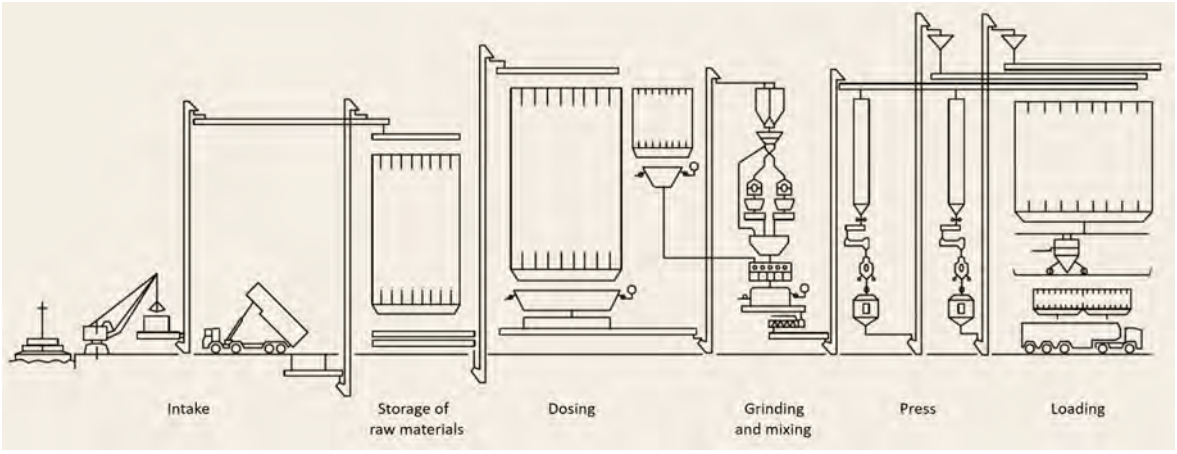


FIGURE 2.1: Schematic overview of plant X with the different areas it includes.

Figure 2.1 shows how an animal feed plant looks like. The manufacturing process starts with the intake and storage of raw materials. Dosing takes place to obtain the right amount of the raw materials for the production of animal feed. After that, the production of animal feed takes place in which the raw materials could be ground, mixed and pressed. Finally, all end-products are stored in silos from which the right amount is dosed and loaded in trucks.

2.1 Overview of the manufacturing process

Plant X produces three forms of animal feed: mash, crumble and pellets, see Figure 2.2. To produce these products, plant X contains four different production line types: two roller lines, one grinder-mixer line (GML), four parallel press lines (PLs) and finally three bulk lines. What these production lines do, is described in detail below. See also Figure 2.4 for a schematic overview of the GML and the four PLs.

FIGURE 2.2: Animal feed forms: mash, crumble and pellets. [3]

The overall process starts with the intake of the several raw materials which are transported to silos. Some raw materials need to be rolled in a pre-processing step.

The roller lines are used for this. All materials will be stored in silos so they are available for further production. This is visualized by the raw material silos in Figure 2.3.

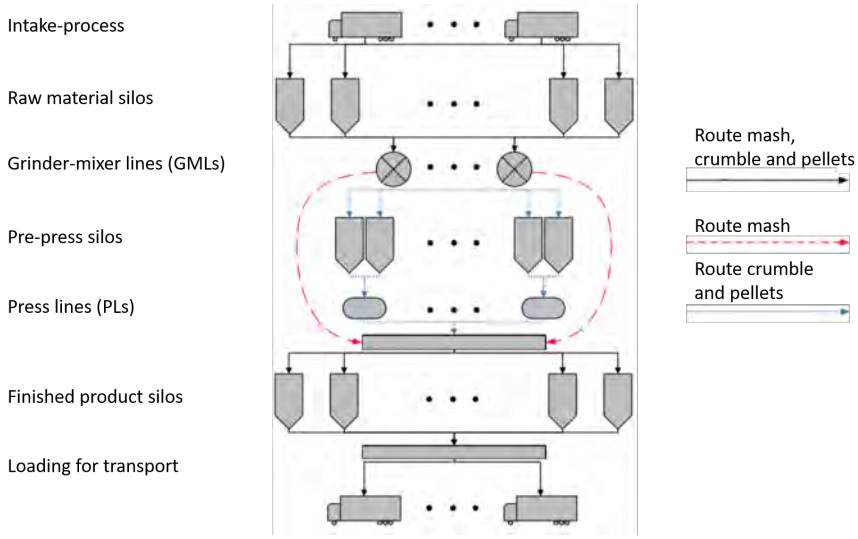


FIGURE 2.3: Abstract visualization of the manufacturing process.

The first step in the production of animal feed consists of grinding and mixing. This happens at the grinder-mixer line (GML). From the silos, the desired weight of the raw materials is weighed and they fall into a bunker (BU5), before they are ground in the hammer mills (HA1 and HA2) in the first production step of the GML, see Figure 2.4A. The ground materials fall into the bunker below the hammer mills (BU6). The second production step of the GML is the mixing phase (NM1), in which the ground materials could be mixed with fine raw materials or liquids. The mixed materials fall into the bunker (BU7) below the mixer. Finally, the product is transported. In this transportation step, there is the possibility to add some liquids (e.g., molasses). These liquids are added while the product is passing (MX1). The reason for the addition of these liquids in this step and not in the mixing phase is that they can be sticky.

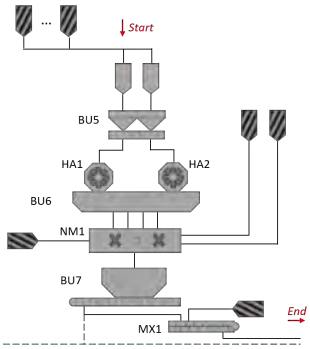
The second step in the production of animal feed is pelleting. This happens at the press line (PL). Mash products do not have to be pressed and are therefore directly transported to the finished product silos. Only the intermediate products for crumble and pellets are processed at a press line, see Figure 2.3.

There are four parallel press lines available in plant X, which are all shown in Figure 2.4. The press lines differ slightly from each other, and therefore it is possible that some products cannot be produced at every press line.

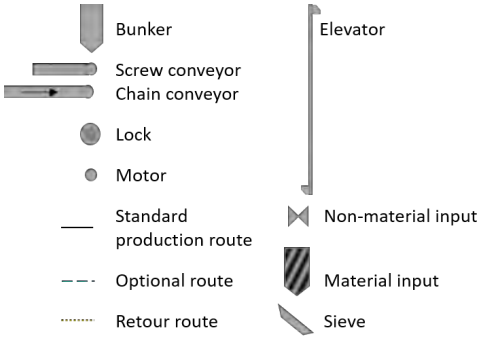
All press lines contain two pre-press silos in which the intermediate products from the GML are stored after transportation. From the pre-press silos, the materials are transported by the press elevator to the press bunker (PB-01, PB-02, PB-03A or PB-03B, and PB-04). After that, the materials are heated using steam (MX-01, MX-02, MX-03A or MX-03B, and MX-04). Press line 2 is the only press line with a BOA (BOA-02). The BOA destroys enzymes using a pressure of normally around 60 to 80 bar.

The next step is to actually press the materials. This happens in a press unit. In PL 2 and PL 3, the materials are pressed by one press unit (PO-02, and PO-03A or PO-03B). In PL 1 and PL 4, two press units can be used in series (PO-01A and PO-01B, and PO-04A and PO-04B). The diameter of the pellets is determined by the size of the compressing holes in the die that is used in the pellet mill.

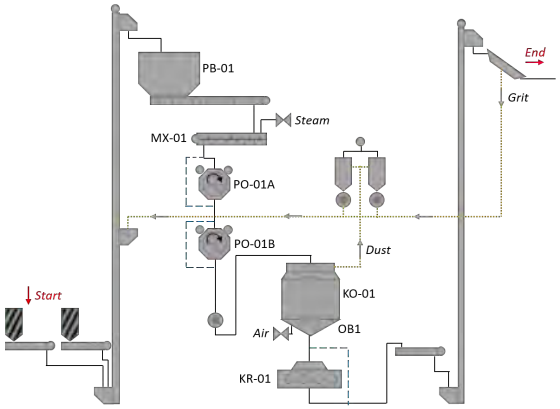
The pressed materials need to be cooled, which happens in a cooler (KO-01, KO-02, KO-03, and KO-04). The materials are cooled by blowing air in the opposite direction (from



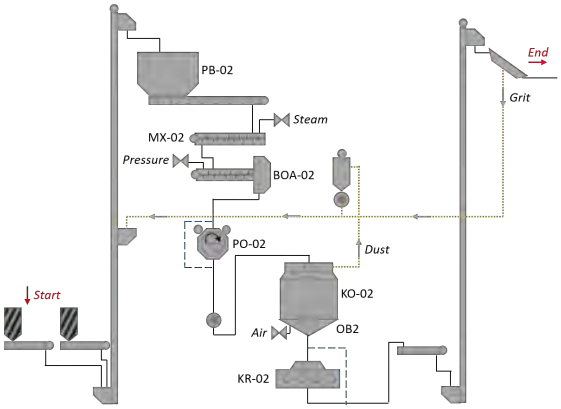
(A) Grinder-mixer line



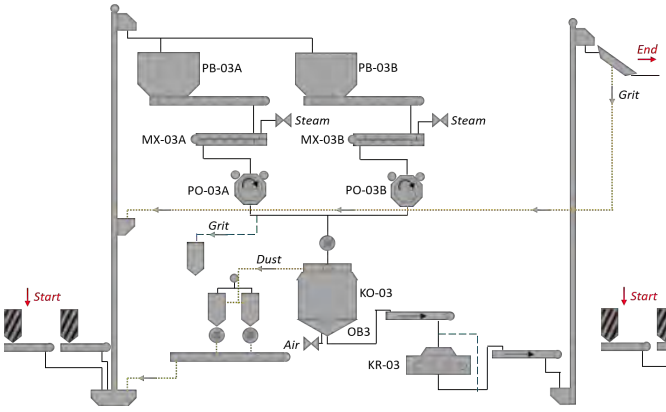
(B) Legend



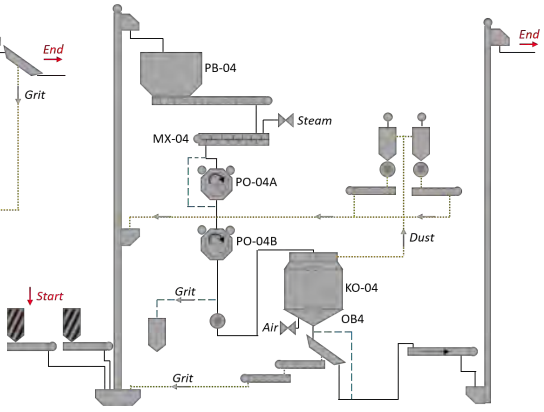
(C) Press line 1



(D) Press line 2



(E) Press line 3



(F) Press line 4

FIGURE 2.4: Abstract visualization of the production lines of plant X.

down to up). This could result in dust, which is caught in the fan and returned to the press elevator, so it can be reused. There is a small difference in the cooler of PL 3 compared to the others; KO-03 does not have an intermediate floor. Therefore, the press units at PL 3 have to wait for KO-03 to be finished, in contrast to the other press lines where pressing can start when the first floor of the cooler becomes empty. The cooling process can then continue on the second floor. When the pellets are cooled, they fall into the bunker below the cooler (OB1, OB2, OB3 and OB4), so the cooler becomes available again.

Except for PL 4, there is the possibility to crumble the pressed materials in a crumbler (KR-01, KR-02 and KR-03). This is the production step where the difference between crumble feed and pellets is made. Finally, the products are transported by a pellet elevator to the final product silos. In the meantime, there is the possibility to filter out too fine parts, also called grit, and return it to the press elevator, so the materials can be reused.

Finally, the end-product will be stored in final product silos, until the truck arrives and transports the product to the client. The bulk lines are used to retrieve the right amount of product for the client out of the silos.

This study focusses on the grinder-mixer line and the press lines because the other production lines are not found to be the bottleneck in the production schedule of plant X.

2.2 Orders

After the introduction to the production process of animal feed, we will give an introduction to the buyers of animal feed. Most customers of plant X are farmers. Nowadays, farmers are increasingly demanding tailored products, a trend known as *mass-customization* [1]. This results in many different products to produce.

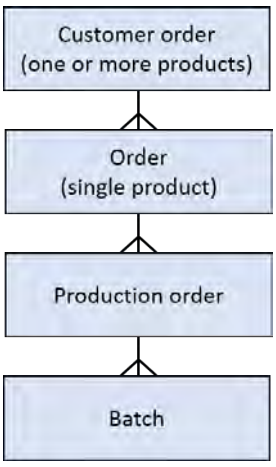


FIGURE 2.5: Hierarchical structure of orders.

A customer (farmer) can order multiple products at a time. This results in a customer order (CO). An order is created for each product in the CO, see Figure 2.5. Since one order needs to be processed at different production lines, we introduce the concept of a production order (PO): an order for a specific production line.

The process at the GML and PL differ in type. The process at the GML is a batch process, while the process at a PL is defined as a continuous process. The GML works in steps, in which the production takes place at one location and is transported to the next location for the next production step. The quantity of product that is processed in a single step of the production line is called a *batch*. For example, if a customer orders 10,000 kg of a product, but the hammer mill in the first step of the GML can only handle 5,000 kg at a time, the production order will be split into two batches of 5,000 kg. Note that the size of the batches depends both on the PO and the capacity of the machines. Therefore, the size of a batch is typically not constant.

In contrast, at a PL the product is processed "on the fly": it is a continuous process. In general, plant X would like to produce the full size of the production order continuously, without interruption. This would result in one "batch" per production order. Therefore, one PL batch could contain multiple GML batches.

Besides the production for customer orders, plant X can also produce stock products. The current scheduling algorithm focusses on customer orders. However, stock production orders can be processed as customer orders, by adding a due-date for which the stock products should be produced.

2.3 Industrial automation

The whole manufacturing process is automated, which means that the whole process is modelled and controlled by a computer. A *batch engine* controls how each production order is produced. The batch engine consists of a physical model, a procedural control model, a recipe model and a production planning. The first three can be described by the ISA-88 framework. ISA-88 is a process control standard for describing the manufacturing process, both in terminology as in modelling. By using ISA-88, the colloquial language in the automation of industrial plants is standardized.

2.3.1 Physical model

The physical model is used to divide the industrial plant into smaller parts based on location and function. Figure 2.6 shows the structure of the ISA-88 physical model. The highest level is *enterprise*, which is the organization that coordinates one or more sites. A site is a physical, geographical, operational or logical subdivision of the enterprise [4]. For example, a customer of ENGIE Industrial Automation (enterprise) with multiple industrial plants at different locations (sites). In general, ENGIE Industrial Automation automates enterprises with only one site. Therefore, the *site* is the highest level that is implemented by ENGIE Industrial Automation.

A site may consist of one or multiple areas, which are defined by a physical, geographical, operational or logical subdivision of the site [4]. In our case, we can divide plant X into six different areas: the intake area, the storage area of the raw materials, the dosing area, the grinding and mixing area, the press area and the loading area, see Figure 2.3 and Figure 2.1.

An area may consist of one or multiple process cells. A process cell is defined as a logical group of equipment that should work together to serve an identifiable processing purpose to process one or multiple batches [4]. For example, a single press line in the press area is a process cell.

A process cell may consist of one or more units, equipment modules and/or control modules. A unit is a collection of equipment and/or control modules that are required to perform one or more major processing tasks [4], such as a hammer mill in the GML (e.g., HA1 in Figure 2.4A) or a press unit in one of the PLs (e.g., PO-01A in Figure 2.4C).

An equipment module is defined as a collection of functional equipment that is required to perform one or more minor processing tasks or process actions [4]. An example of an equipment module is the collection of equipment that is used to change the sieve of a hammer mill.

A control module is the lowest level of equipment in the physical model. It is defined as (a collection of) equipment that is used to perform a basic control function [4]. It is not

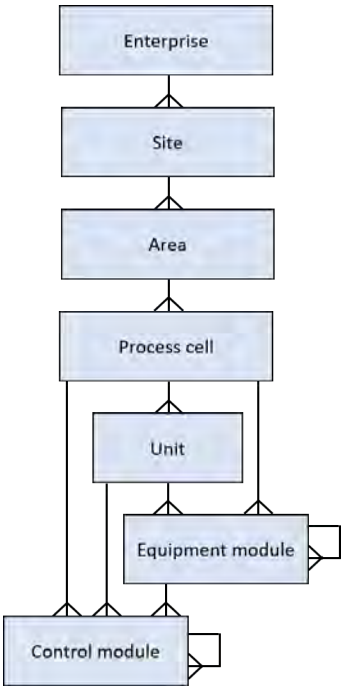


FIGURE 2.6: Structure of the ISA-88 physical model.

able to perform procedural logic. Control modules are often building blocks that directly drive the input/output (I/O) device that communicates between the information system, e.g., computer, and the physical system. An example of a control module is a valve that can be opened/closed or an empty detector that detects if a bunker is empty or not.

2.3.2 Procedural control model

The procedural control model is a multi-level hierarchical model to accomplish the task of a complete process (or part of a process) based on the resources of a specific process cell [4]. The structure of the procedural control model is shown in Figure 2.7. The ultimate goal of the hierarchy is to efficiently organize the process-oriented tasks that are to be executed (either automatically or manually) at its lowest level [4].

The highest level of the procedural control model is the procedure. The procedure defines the order of processing for an entire batch. It may specify the execution order of one or more unit procedures, which may be in series, in parallel, or a combination of both [4].

A unit procedure specifies a contiguous production sequence that takes place within a single unit [4]. For example, everything that needs to be done at a hammer mill: supplying, sieve switching and grinding.

A unit procedure may consist of one or more operations. Each operation specifies a major processing sequence, usually to take material being processed from one state to another involving a chemical or physical change [4].

A phase is the lowest level procedural element in the procedural control model and is intended to accomplish all or part of a process action [4]. It is linked to an equipment module in the physical model. Examples of phases are sieve switching, grinding and mixing.

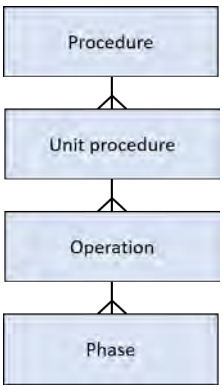


FIGURE 2.7: Structure of the ISA-88 procedural control model.

Since the unit procedures of plant X consist of only a few phases, ENGIE Industrial Automation does not consider the concept of operations in its procedural control model.

2.3.3 Recipe model

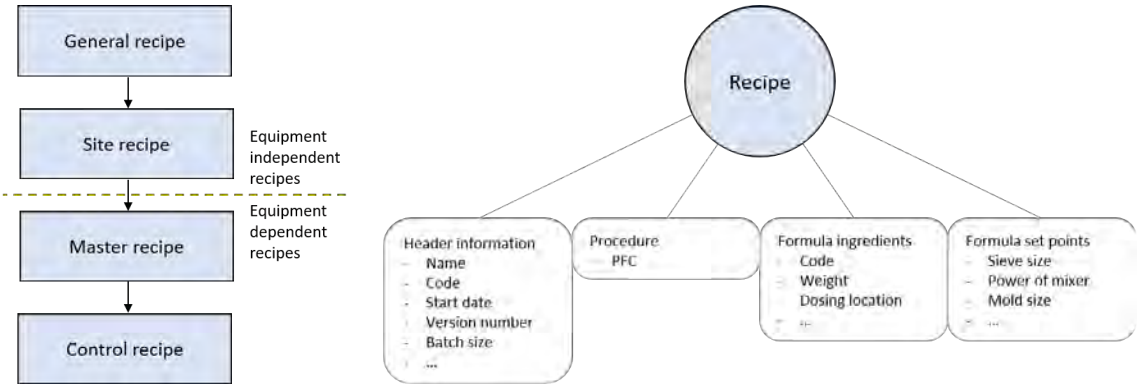
The last model described in the ISA-88 process control standard is the recipe model. Recipes provide a way to describe products and how those products are produced [4]. There are four recipe types: the general recipe, the site recipe, the master recipe and the control recipe, see Figure 2.8A.

The general recipe serves as the basis for lower-level recipes. It contains information about nutritional requirements. This recipe is applicable at the enterprise level and is independent of the specific site. It is created without specific knowledge of or information about the process of cell equipment that will be used to manufacture the product [4].

The site recipe is specific to a particular site. It may accommodate specifications like the language, the policy or locally available raw materials [4]. This recipe is still independent of the equipment.

The master recipe is created to produce one or more batches. It is specific to the equipment, raw materials and the capabilities of a process cell or a subset of process cell equipment [4]. A master recipe is required for the creation of a master control recipe.

A control recipe serves as the production instructions of a unique batch. It is initiated by the master control recipe, but it may contain modifications that can be made at any time,



(A) Recipe types (B) Recipe contents

FIGURE 2.8: Structure of the ISA-88 recipe model.

for example, to account for actual raw material quantities, material properties, the selection of units, or appropriate sizing [4]. Since modifications of a control recipe can be made over a period of time based on schedule, equipment, and operator information, a control recipe may go through several modifications during the batch processing.

Each recipe consists of header information, the procedure, the ingredients and the setpoints, see Figure 2.8B. The header information contains the product name, version and code, when the recipe is used and how much of the product is produced. The procedure contains the PFC (procedural function chart), which can be thought of as a drawing of the physical process, in which is described when each equipment module is allocated and the corresponding phase is started, and what happens afterwards. The formula of the ingredients may contain multiple ingredients, of which it specifies their codes and how much of them is required. Finally, the formula of the setpoints contains the settings of the required equipment modules. For example, the power of the mixer, the size of the holes in the sieve of each hammer mill and the power of the press unit.

2.4 Production durations

The goal of this research is to accurately estimate the duration of the production processes in plant X. Since these estimations will be used in the scheduling algorithm, these estimations should comprehend the time that equipment is occupied for a specific production step. This includes the time that is used for transportation. However, we do not expect that there is much spread in these transportation durations.

The scheduling algorithm used by ENGIE Industrial Automation in plant X does not take the intake-process, and the loading process of trucks into account yet. It specifically focusses on the grinding and mixing area and the press area. Which means that the scheduling algorithm chooses which press line to use for which production order, and schedules the order of the batches in the GML. The production durations of the GML are visualized in Figure 2.9.

The first production duration, (1) in Figure 2.9, we would like to estimate is the time it takes to grind the raw materials in the hammer mills. Since the two hammer mills in the GML are used simultaneously, it was chosen to estimate the duration of this step by the production duration of the left hammer mill (HA1). The difference in the start time of the supply conveyor and the end time of the grinding phase represents this duration, because the supply starts when the hammer mill is allocated and the hammer mill is deallocated after completion of the grinding phase. The grinding phase completes when all materials fit

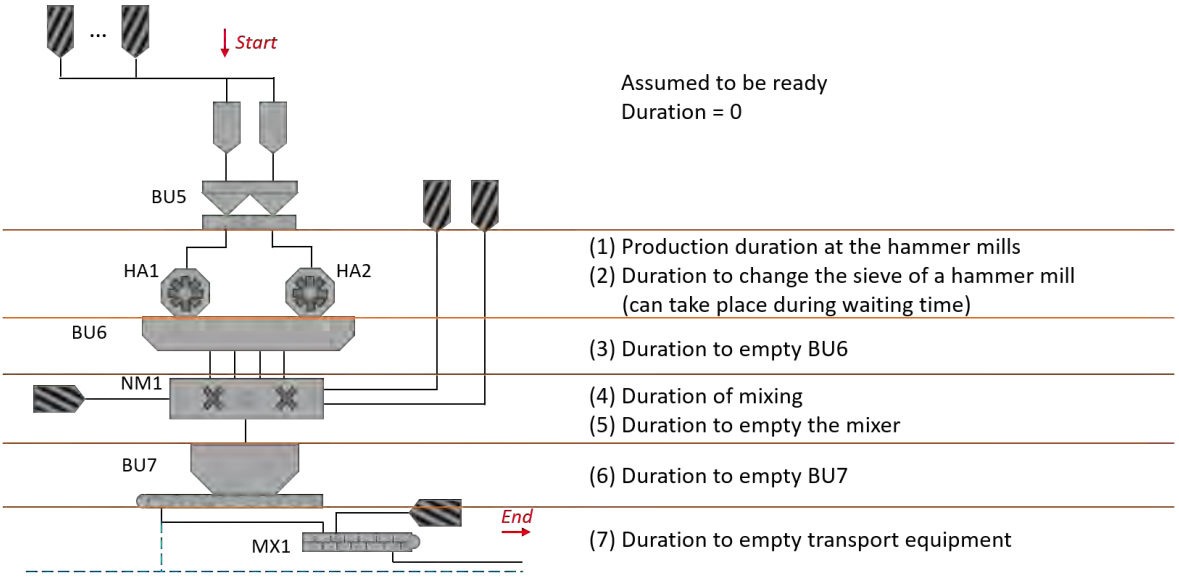


FIGURE 2.9: Production durations of the GML.
The legend is shown in Figure 2.4B.

through the sieve of the hammer mill and fall into the bunker beneath it (BU6). This process is improved by creating a vacuum between the hammer mill and BU6. Therefore, the hammer mill and BU6 are allocated simultaneously by the scheduling algorithm. The duration of this step will further be denoted by HA1_TOEV ("toevoeren" is the Dutch translation of supplying).

The second duration, (2) in Figure 2.9, is the time it takes to switch the sieve of the hammer mill. This duration is only relevant if the batch requires another sieve size as the previous batch, and can be performed during waiting time. The sieve switch is automated, so we do not suspect to see much differences in the sieve switching durations. Since we focus on only the left hammer mill (HA1), we will also focus on the time to change the sieve of the left hammer mill (HA1). This duration will be referred to as HA1_ZEEF (the Dutch word for sieve is "zeef").

After the hammer mills are finished, the bunker beneath it (BU6) will be emptied. The time to empty the bunker is the third estimated duration, see (3) in Figure 2.9. This duration is calculated by taking the time the valve of the bunker is open, with four additional seconds for the deallocation of the hammer mills. This duration will further be denoted by BU6_LOS ("lossen" is the Dutch word for unloading).

While unloading the bunker, the batch falls into the mixer (NM1). The time it takes to mix the materials of the batch, (4) in Figure 2.9, is obtained by taking the start and stop time of the mixing phase, further referred to as NM1_step. Note that this duration includes the time to dose the right amount of liquids from the liquid silos.

The fifth duration, (5) in Figure 2.9, is the time it takes to empty the mixer. There are two reasons to estimate this duration independently from NM1_step. First of all, the duration to empty the mixer is more constant than the mixing duration and the durations likely depend on other variables. Secondly, the valve must wait to open until the bunker beneath it is empty. So, there might be waiting time in between. The time to empty the mixer is further referred to as NM1_Los_step.

Since it can take some time to transport a batch to the next destination, another bunker (BU7) is included so the mixer can already become available for a new batch. When the transportation of the previous batch finishes, the bunker (BU7) can unload into the transportation system, see (6) in Figure 2.9. The time it takes to become completely empty could

not be obtained by the unloading step of the bunker, because this happens to be inaccurate. The difference in the finishing time of the addition of liquids like molasses (MX1) and the starting time of the transportation equipment is used instead. However, the bunker becomes earlier available than MX1 finishes. The time that the bunker is earlier available than MX1 was accurately estimated by 11 seconds, and thus we could easily correct for this. The time it took to empty the bunker (BU7) is further mentioned as MML_AFV_TR ("MML" is the Dutch abbreviation for the grinder and mixer line, "afvoeren" is Dutch for discharging and "TR" stands for transportation).

The last duration of GML we would like to estimate accurately is the time it takes to transport a batch to the next destination, see (7) in Figure 2.9. Off course, this duration depends on the location of the destination. This duration can only be observed in the data when the next batch is of another product (if not, the batches are transported continuously), because plant X has a transportation system that can switch the destination "on the fly" ("vliegend-om" in Dutch). Some orders are called "special", which means, in practice, that the product cannot be stored in the same place as other orders for the same product. In general, the reason for becoming special is that there is a (small) change in the recipe, for example, the addition of medicines. If the order is special, the product is seen as different from other orders.

If the product of a batch is different from the previous batch, the batch has to wait for the transportation equipment to become completely empty to avoid mixing. The time it takes to empty the transportation equipment is calculated by taking the difference between the finishing time of the transportation equipment and the moment that the bunker (BU7) becomes empty. As mentioned before, the time at which the bunker becomes empty is given by the finishing time of MX1 minus 11 seconds. The time to empty the transportation system is referred to as MML_AFV_TR_NADRAAI ("nadraai" can literally be translated to "afterwards rotating", which refers to the rotation of a conveyor).

In contrast to the GML, a PL concerns a continuous process. Since we assume that the transportation to the final product silos is not a bottleneck in the system, the last unit of a press line that is taken into account in the scheduling algorithm is the cooler. The deadlines of the orders are therefore brought forward in the scheduling algorithm to make time available for the loading of the truck.

The coolers of PL 1, 2 and 4 contain an intermediate floor, with as result that the press unit can already start before the cooler finishes. Therefore, the cooling durations of these press lines can be ignored too. Only the cooling duration of PL 3 is relevant because the press unit of PL 3 has to wait for the cooler to finish.

The pressing duration is calculated by taking the difference between the start and stop time of the unit procedure of the press unit. This results in four production durations, for each production line one: PL_PO1, PL_PO2, PL_PO3 and PL_PO4. The production duration of the cooler at PL 3 is calculated by taking the difference between the stop time of pressing and stop time of the cooling phase of the cooler. This duration is referred to as KO3_idle_time.

An overview of all production durations and the current approach of estimating them are given in Table 2.1. The goal of this research is to accurately estimate these production durations. Therefore, these production durations are the *target variables* of this research.

	Target variable	**Line**	**Description**	**Benchmark estimator**
1	HA1_TOEV	GML	Production time of the grinding step.	Median duration per 1,000 kg of last 5 batches of the same article.
2	HA1_ZEEF	GML	Sieve switching time of a grinder.	Median over all sieve switching times.
3	BU6_LOS	GML	Time it takes to empty the bunker under the grinders.	Median duration over all batches of the same article plus 4 seconds to correct for time between HA1_ZEEF and BU6_LOS.
4	NM1_step	GML	Production time at mixing unit.	Median duration per 1,000 kg of last 21 batches of the same article.
5	NM1_Los_step	GML	Time it takes to empty the mixing unit into the bunker.	Median over all batches.
6	MML_AFV_TR	GML	Duration to empty a batch from the bunker under the mixing unit.	Median duration per 1,000 kg over all batches of the same article.
7	MML_AFV_TR_NADRAAI	-	Transport time from the bunker under the mixing unit to destination.	Median duration over the last 31 batches with the same destination.
8 9 10 11	PL_PO1 PL_PO2 PL_PO3 PL_PO4	PL	Production time at the press unit of respectively PL 1, 2, 3 and 4.	Robust regression estimation[a] to relate batch size to duration per article for each feasible PL.
12	KO3_idle_time	PL	Time to empty the cooler after the press unit at PL 3.	Median duration of last 31 batches.

TABLE 2.1: Overview of the different durations to estimate.

[a] `https://www.statsmodels.org/stable/generated/statsmodels.robust.robust_linear_model.RLM.html`

3 Literature

This section discusses some relevant literature in the estimation of production durations for scheduling purposes.

Until the late 1990s, the majority of the literature assumed deterministic production durations [5], which are often estimated by taking the average over the production durations of the same product type. For example, Toso et al. (2009) assume the same production times per batch for products within the same product family [6].

However, in practice, production durations are typically not constant and, in general, uncertainty in the production durations makes a scheduling algorithm less accurate. Scheduling under uncertainty is a field of study in which these kinds of uncertainties are taken into account in the scheduling algorithm. This field can be split by three different descriptions of the uncertainty: *fuzzy scheduling*, *stochastic scheduling* and *robust scheduling* [5].

In stochastic scheduling, uncertainties like the variability in the production durations are incorporated in the model by introducing random variables for them. For these random variables are particular processing time distributions assumed, which should comprehend the randomness in the production durations. For example, the exponential distribution is frequently used for this purpose [7].

Robust scheduling focusses on the creation of a schedule which is *robust*. According to Goren and Sabuncuoglu (2008), a schedule is robust if its performance "does not significantly degrade in the face of disruption" [8]. Therefore, robust scheduling aims to find the best schedule in the worst-case scenario [9].

The last approach in scheduling under uncertainty is fuzzy scheduling. "Fuzzy scheduling is based on the claim that probability distributions cannot be estimated correctly most of the time and therefore it models the imprecision of input data by using fuzzy numbers rather than probability distributions" [8]. For example, Yao and Lin (2002) represented job processing times by interval-valued fuzzy numbers [10].

As mentioned, most literature focusses on the incorporation of the variability in the scheduling algorithm by using fuzzy, stochastic or robust scheduling. However, relatively little research has been done to the predictability of this variation of the production durations.

Most of the research in this area has been done to the effect of deterioration and/or learning phenomena on production durations, which focusses on the idea that "in many industrial settings, the processing time of a job changes due to either job deterioration over time or machine/worker's learning through experiences" [11]. Biskup (1999) firstly introduced the concept of the learning effect into scheduling problems [12] and Gupta and Gupta (1988) firstly introduced job deterioration into scheduling [13]. Later, these two phenomena were combined "because the phenomena can be found in many real-life situations" [11]. For this purpose, models were created which consist of a mathematical formula that should comprehend these phenomena [11].

F. Matsunari et al. (1996) made use of more variables in their "process time estimating apparatus" [14]. The "process time estimating section" of this apparatus aims to estimate

the production durations by using a neural network. The apparatus is applied in a metal die manufacturing plant. First, drawing information (CAD) is transformed into other variables, e.g., size. After that, these product variables and the machine settings of the current production step are selected. With this information, a neural network was trained. Every time a new duration is measured that deviates less than 20% with the predicted value, it is assumed that this record is not related to a machine failure (or another kind of outlier), and thus can be used to update the neural network. So this apparatus is designed to keep learning.

F. Matsunari et al. claim to accurately estimate the production durations in the metal die manufacturing plant. Therefore, it would be interesting to investigate if a neural network would also be able to accurately estimate the production durations of the specific industrial plant of this research.

Neural networks and other machine learning algorithms show many successes in many industries. Jiang et al. (2017) show an increase in the number of articles about deep learning in healthcare and disease category on PubMed [15], Spiegeleer et al. (2018) showed how machine learning could be used for quantitative finance [16], and Antoniou and Koutsopoulos (2006) estimated traffic dynamics models using machine learning methods [17].

In the industrial manufacturing sector, an increase in the usage of machine learning algorithms can be observed too. This trend is part of the *fourth industrial revolution* or *Industrial 4.0* [18] in which a modernization of the manufacturing industry takes place by adapting recent advances in the ICT [19]. Fault detection is an example of such a machine learning application, which is subject of many recent studies [19], [20]. In this context, more and more data of industrial plants are stored. This offers possibilities for data science.

It would be interesting to not only implement a neural network for the estimation of the production durations, but also to compare the results with the results of other machine learning algorithms that already showed good results for other applications, but are not used for the estimation of production durations in literature before. Note that the study of F. Matsunari et al. in which neural networks were used, was performed in 1996. Therefore, it is interesting to see if more recent models are able to outperform the neural network. For example, LightGBM and XGBoost are models that are often used in on-line machine learning competitions, for example, *Kaggle*. "Among the 29 challenge winning solutions published at Kaggle's blog during 2015, 17 solutions used XGBoost" [21]. In this research, some modern and/or frequently used machine learning models will be used and compared. These models will be described in Section 7.

4 Data

The data that is used to estimate the production durations, is described in this section. This section starts with a description of the data collection process. After that, an explanation is given about which filters are used and how many records are removed because of them.

4.1 Collecting data

All data used for this research is provided by ENGIE Industrial Automation. The data was obtained by directly querying a SQL-database. There are two databases available: one with a data dump of 2017 (2016/08/14 – 2017/08/14) and one of 2018 (2017/12/14 – 2018/12/17). Note that there is a time gap of four months between the first and second database. All records of both databases belong to the same animal-feed plant; plant X.

There is a lot of data available, and therefore we should be very selective in what to query at the same time, to prevent memory errors. This resulted in several SQL-queries. First of all, the orders were obtained. One order contains the information of which article is produced. From each order, the corresponding production orders can be obtained, which contain the information of which production lines are used for the production of the article.

After that, all batches at the GML were queried and combined with the production order information. The same is done for the batches at the PLs.

Finally, all production durations of the batches are queried separately, to which we can add the batch information of the corresponding production line.

The setpoints were obtained for each production step and are merged with the production durations for which the setpoints are relevant. For example, the setpoints of the press units are relevant for the production durations at the press units (PL_PO1, PL_PO2, PL_PO3 and PL_PO4) and not for the production durations at the hammer mills (HA1_TOEV and HA1_ZEEF).

The ingredients were obtained the same way as the setpoints. There are three locations where ingredients could be added in the process (see material inputs in Figure 2.9). The first is the input of the hammer mills. These ingredients are relevant during the full production process. In the mixer, other ingredients could be added. Since only the liquid dosing is part of the mixing duration, only the liquid ingredients are potentially relevant for the prediction of the mixing duration (NM1_step), and all ingredients are potentially relevant for the production steps after the mixer. Finally, (sticky) ingredients could be added during transportation, which are possibly relevant for the predictions after the transportation; PL_PO1, PL_PO2, PL_PO3 and PL_PO4.

Since the temperature and humidity may affect the production durations, weather data for each day between the 14th of August in 2016 and the 17th of December in 2018 was downloaded from the website of the KNMI[1] for the nearest weather station of plant X.

[1] https://www.knmi.nl/kennis-en-datacentrum/achtergrond/data-ophalen-vanuit-een-script

4.2 Cleaning batches

Querying all production orders from the database results in 40,832 unique orders and 103,436 unique production orders in the 2017 database, and 40,446 unique orders and 108,521 unique production orders in the 2018 database, see Table 4.1.

4.2.1 Orders and production orders

To obtain the relevant orders for this research, some filters are applied. First of all, only orders are selected which contain at least the first step of the GML (the supply to the hammer mills), because we are not interested in orders that only contain production orders for the rolling lines and bulk weighers. Approximately a quarter of the orders is removed by this filter, see Table 4.1.

	2017		2018		Total	
Filters	**Orders**	**Production Orders**	**Orders**	**Production Orders**	**Orders**	**Production Orders**
-	40,832	103,436	40,446	108,521	81,278	211,957
1	29,546	91,478	31,213	98,573	60,759	190,051
2	29,546	90,025	31,213	97,328	60,759	187,353
3	29,540	90,015	31,213	97,328	60,753	187,343
4	29,537	57,201	31,213	61,115	60,750	118,316
5	29,519	57,183	31,199	61,101	60,718	118,284

TABLE 4.1: Overview of the number of (production) orders after applying each filter.

The second filter ensures that all considered production orders contain a minimum of one finished batch, and the third filter removes orders that were changed after the data dump date because these orders are considered as unreliable.

Finally, only production orders for the GML and the press lines are selected by applying the fourth filter. The removed production lines are not relevant to this research. For the sake of completeness, orders which suggest going to a press line after the completion of the GML must contain a production order for that press line. This is ensured by the fifth filter.

Applying all filters results in a total of 60,718 orders and 118,284 production orders over two years of data, see Table 4.1. The 3,502 orders for mash products contain at least one production order for the GML. The 57,217 orders for crumble and pellet feed contain at least two production orders: one production order for the GML and one for a press line. Therefore, the number of production orders of the complete dataset is rough twice the number of orders in the dataset.

Note that there is one order (3,502+57,217=60,719) which was first produced as mash product, and then corrected to pellet feed. This order contains, therefore, two production orders for the GML: one that goes to the end-silos afterwards, and one that goes to the press lines. Since the first production order was already finished before it was corrected, we could still use the observations of the production durations of this order.

4.2.2 Batches

Three filters are applied to the individual batches corresponding to the obtained production orders. The first filter ensures the batches to be finished by selecting completed batches (state=3). The second filter filters out all rejected batches (rejected=0). The last filter requires the batch to have a batch size of more than zero kilograms.

As mentioned, one order corresponds to one or multiple production orders, and one production order corresponds to one or multiple batches, see Figure 2.5. As mentioned, the 60,718 orders from 2017 and 2018 can be split into two groups: 3,502 orders belonging to mash products, and 57,217 orders belonging to crumble and pellet feed. All orders are produced at the GML. To these 60,718 orders belong 60,749 GML production orders. The batches are obtained for these production orders, which results in 127,705 GML batches.

The mash products are not produced at the press lines. The 57,217 orders for crumble and pellet feed correspond to 57,535 press line production orders. The batches are obtained for these production orders, which results in 57,536 press line batches. Note that the process at the press line is a continuous process. Therefore, the number of production orders and batches are approximately the same.

4.2.3 Batch sizes

The batch sizes at the GML are used as a global unit in the scheduling algorithm of ENGIE Industrial Automation. It is assumed that the requested and produced amount are approximately the same. However, when something went wrong during the production process, for example, a significant deviation could occur. Therefore, we require the produced amount of kilograms to be approximately the same as the requested batch size. To define how similar the produced amount and requested amount should be, the interquartile range (IQR) rule is used, which was introduced by J. Tukey in 1977 while he was inventing the boxplot [22].

IQR rule The IQR rule is a rule of thumb for separating outliers from a dataset by defining some boundaries, called Tukey's fences [22], between which the data points are seen as 'normal' and outside which the data points are seen as 'outliers'. The IQR rule first defines the interquartile range (IQR), which is equal to the difference between the first (Q1) and third (Q3) quartile, see Figure 4.1, and therefore consists of the 'centre' 50% of the data. Using the IQR, the lower bound is defined as the first quartile minus k * IQR, and the upper bound as the third quartile plus k * IQR [22]. Generally, k is set to 1.5 for detecting outliers. The reason for 1.5 is that when the IQR rule is applied to a standard normal distributed dataset, the percentage of the data points that is seen as an outlier is approximately 1%. However, when there is more variation in the dataset and k=1.5 results in too many outliers, k is often set to 3, which indicates data that is "far out" [22]. See Figure 4.1 for a visualization.

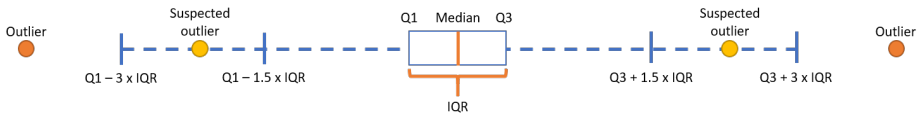


FIGURE 4.1: Visualization of a boxplot with outlier boundaries k=1.5 and k=3 based on the IQR rule of J. Tukey [22].

Now, we could calculate the percentage difference between the produced and requested batch size, which could be seen as some tolerance on the batch size, where 0% means that the produced and requested amount are the same. The IQR rule could then be used to separate the 'normal' tolerances from the outliers, by using k=1.5 because the tolerances are approximately normal distributed, see Figure 4.2.

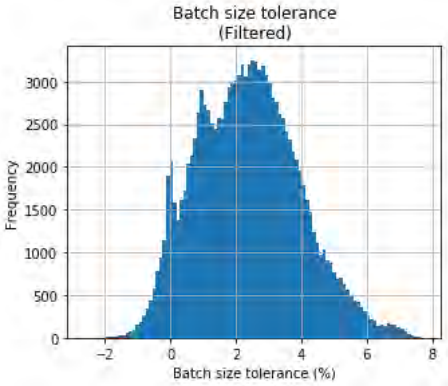


FIGURE 4.2: Histogram of the batch size tolerance: the percentage difference between the produced and requested batch size.

However, the dosing procedure in plant X could be changed in time, which could change the batch size tolerance distribution. Therefore, we prefer adapting outlier detection above the usage of fixed boundaries. An adaptation of the IQR rule of J. Tukey is introduced in this research to accomplish this, which is called the windowed IQR rule. In this method, the IQR rule of J. Tukey is applied in a 'windowed' fashion. The next batch, say batch A, is detected as outlier if it is detected as outlier by the IQR rule, in which only the most recent B batches are taken into account that were produced up and until batch A. In this way, the outlier boundaries respond to changes in the underlying distribution that can occur over time.

For the selection of this window size B, the numbers 50, 100, 500, 1,000, 5,000, 10,000, 50,000, 100,000 and the total number of batches (127,705) were tested, and the value was selected with the least number of detected outliers. This approach is taken to limit the number of filtered out batches and to select the window that best 'follows' the distribution of the data points over time. This resulted in the choice of a window size of 10,000 batches, which marks 1,578 batches (1.24%) as outliers. Figure 4.3 shows the tolerance of all batches in the dataset, sorted by their starting date. The tolerances that are marked as outliers by the IQR rule are visualized as red dots, and the accepted tolerances by blue dots.

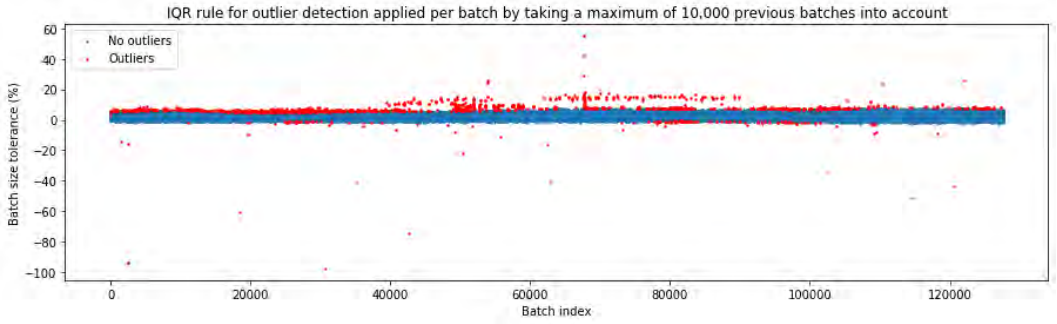


FIGURE 4.3: Batch size tolerance defined as the percentage difference between the produced and requested batch size of GML batches, marked as outlier or not.

As we assume the produced and requested amount to be approximately the same, the detected outliers are removed from the dataset. This results in 126,127 GML batches which belong to 60,306 orders. The resulting histogram of the tolerances is shown in Figure 4.2. Figure 4.2 shows three peaks: one at 0%, one at 1% and one at 2.5%. An explanation can be found in the three product types, see Figure 4.4.

The centre histogram of Figure 4.4 shows the distribution of the batch size tolerance of mash products, which happens to be located around 0%, which means that approximately the same amount was produced as requested.

The produced amount of crumble products is approximately between 1 and 2.5% more than the requested amount, see the left histogram of Figure 4.4. Finally, the pellet products have the highest batch size tolerance, of approximately 2.5%, see the right histogram of Figure 4.4. These percentages suggest that more product is produced than requested. However, this is the result of additional liquids, e.g., water, which evaporates at the press line when the product is heated.

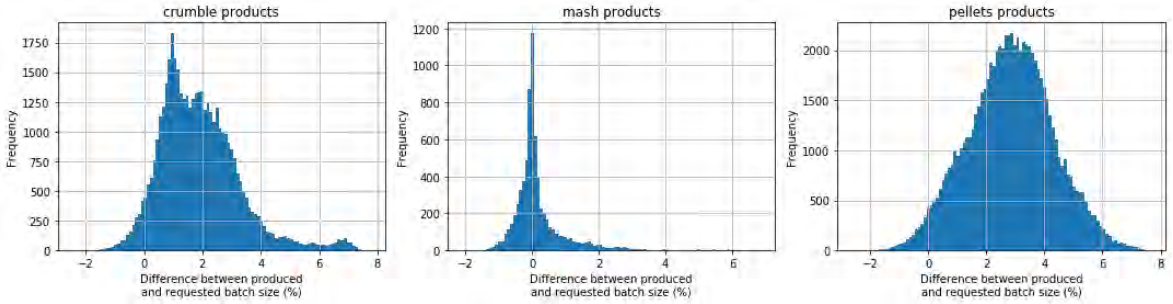


FIGURE 4.4: Difference between produced and requested batch size in percentages of batches at the GML, filtered per article type.

The final remaining number of batches per production line are shown in Figure 4.5. Note that these batch size filters are only applied on the GML batches because the GML batches are the used unit in the scheduling algorithm. The estimation of future PL batches will be performed by the summation of the corresponding GML batches because pressing is a continuous process. Therefore, no such filter is applied to the PL batches. This results in differences in the number of orders at the GML going to the press lines and the orders produced at the press lines.

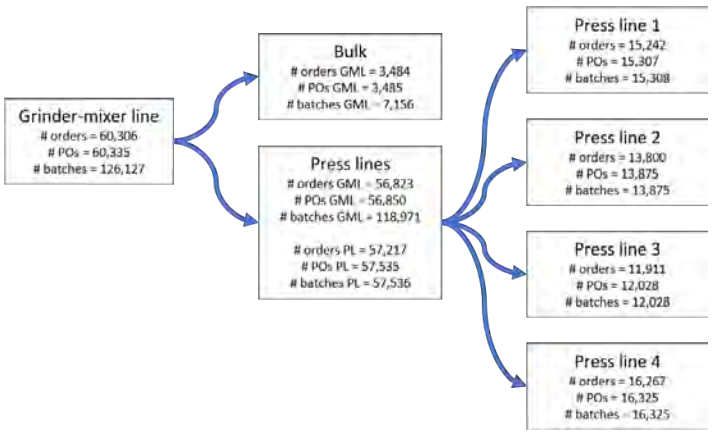


FIGURE 4.5: Orders per production line.

In addition, remember the single order that was first used for the production of mash, and after that was corrected to produce pellet feed. This single order corresponds with the difference of one GML order in the first (60,306) and second step (3,484+56,823=60,307) in the flow diagram shown in Figure 4.5.

4.3 Cleaning production durations

For each batch, the production durations can be obtained by looking at the start and end times of the individual production steps. This results in a distribution of durations for each target variable that is listed in Table 2.1. The final cleaning filters are used to filter out 'abnormalities' in these observed durations of each individual target variable.

First of all, it could happen that, for example, an error occurs during the production process which could not be solved immediately. Such events could result in very long durations. However, the goal of this research is to predict the production durations for the scheduling algorithm, which optimizes the schedule for a 'normal' situation. When something happens, the scheduler could rerun the algorithm after solving the issue, to obtain a new optimized schedule. Therefore, the outliers, mostly occurring above the mean value, should be filtered out.

The shape of the production duration distributions could change over time. For example, when a new article is created which requires some other machine settings which influence the production duration. Therefore, the outlier detection should be able to understand the difference between outliers and the change in the distribution. For this, the IQR rule of J. Tukey [22] (Paragraph 4.2.3) could be used again in a windowed fashion. However, the distributions are overall very right-skewed, and therefore it was decided to use a lower bound of k=1.5 and an upper bound of k=3 in the IQR rule, which limits the number of detected outliers.

For the selection of the window size the values 50, 100, 500, 1,000, 5,000, 10,000, 50,000, 100,000 and the total number of batches (126,127) are tested. The value with the least number of detected outliers is chosen because this window size is likely the best fit for the distribution of the data points over time. However, this does not guarantee the best-selected window size. Therefore, the detected outliers are checked visually, so the selected window size could be changed if needed. The same window size can be used for future batches. However, it is important to keep monitoring which batches are detected as outliers, in order to prevent undesirable changes in filtered batches. This could, for example, be done by notifying experts when a significant increase in the number of detected outliers occurs.

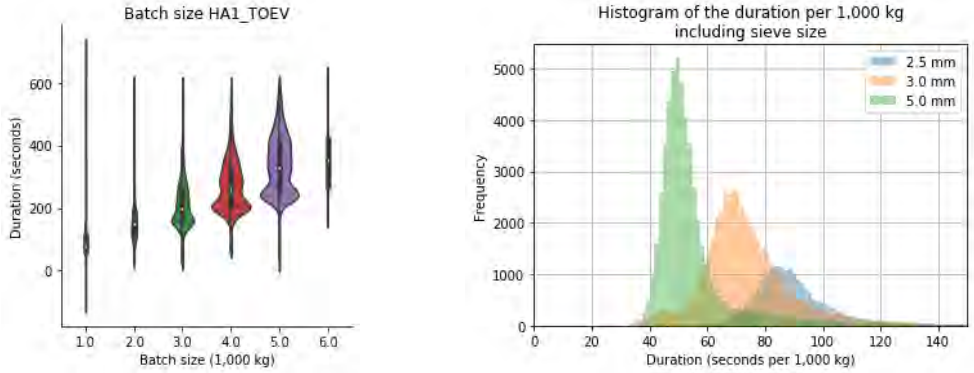
Secondly, the setpoints (machine settings) of batches could be changed during the production step. However, the final model needs to predict the duration before the production step starts, for which the influence of setpoint values of changed batches is not completely representative. Therefore, all batches of which the setpoints are changed after the production step started are filtered out. This is done by looking at the batch related events.

4.3.1 Production durations at grinder-mixer line

The filters are applied to all target durations of the GML. In this way, the target-specific data is obtained which is used to estimate the corresponding durations.

HA1_TOEV

The first target variable is HA1_TOEV, which represents the duration of the supplying and mixing phase of the first hammer mill. Since not always the first hammer mill is used, for example, because of a machine failure, not all GML batches have an observation for this duration. The number of batches with an observation of HA1_TOEV is 125,732 (99.6%).



(A) Violin chart which shows the correlation between the (produced) batch size and the HA1_TOEV duration. Each violin shows the shape of the distribution and the total area of the violin indicates the number of data points.

(B) Distribution of the HA1_TOEV durations per 1,000 kg per sieve size.

FIGURE 4.6: Figures showing correlations between the HA1_TOEV durations and the batch size and the sieve size.

Since the HA1_TOEV duration depends a lot on the batch size and the sieve size, see Figure 4.6, it was decided to apply the windowed IQR rule on the production duration per 1,000 kg per sieve size, instead of the total production duration. The window size with the smallest number of detected outliers was found to be 10,000 for a sieve size of 2.5 mm, the total history for a sieve size of 3 mm (49,906 batches), and 50,000 batches for a sieve size of 5 mm. This results in 4,998 detected outliers (3.97%). These outliers and the other data points are shown in Figure 4.7.

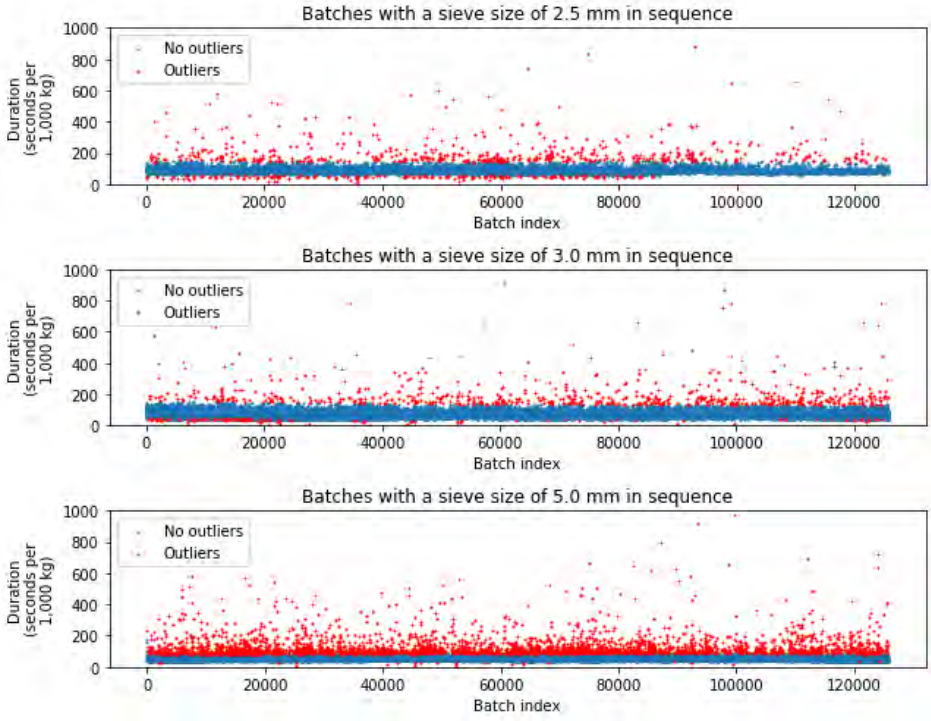
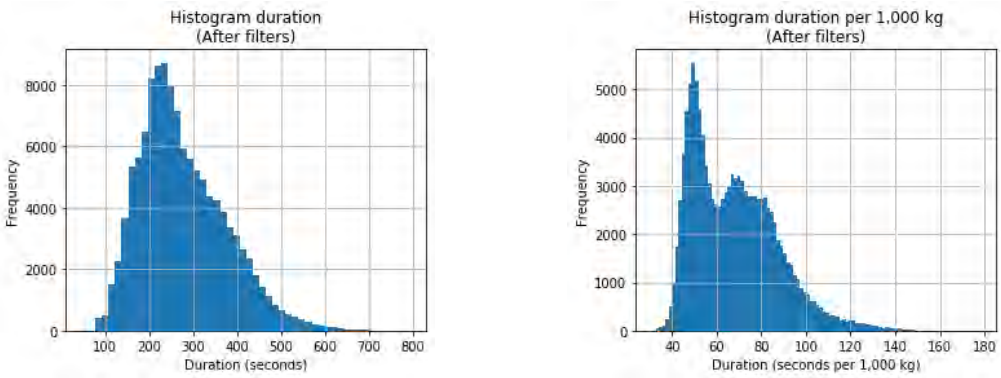


FIGURE 4.7: HA1_TOEV durations sorted by start date, marked as outlier or not.

The outlier filtering results in 120,734 batches with an observation for the HA1_TOEV

duration. From these batches, the batches with a change in the setpoints (the start or maximum capacity of the supply conveyor, the power of the supply conveyor, the speed of the mixer and the sieve size) after the start time are filtered. This filter removes another 207 observations, resulting in 120,527 batches.



(A) Total production duration.

(B) Duration per 1,000 kg.

FIGURE 4.8: The distribution of the HA1_TOEV durations after applying filters.

The Figures in 4.8 show the final distributions of both the HA1_TOEV total durations (Figure 4.8A) and the durations per 1,000 kg (Figure 4.8B) of the 120,527 remaining batches. The duration per 1,000 kg ranges from approximately 31 seconds to 178 seconds with a mean value of 67.8 seconds. This results in a range of 48 seconds to 793 seconds (13.2 minutes) for the total HA1_TOEV duration, with a mean value of 276 seconds (4.6 minutes).

HA1_ZEEF

The second target variable is HA1_ZEEF, which represents the duration of switching the sieve of the hammer mill. When we filter the 126,127 GML batches on the occurrence of a sieve switch and a duration of at least 5 seconds (the maximum time to register that no sieve switch is needed), 44,897 batches (35.6%) are left.

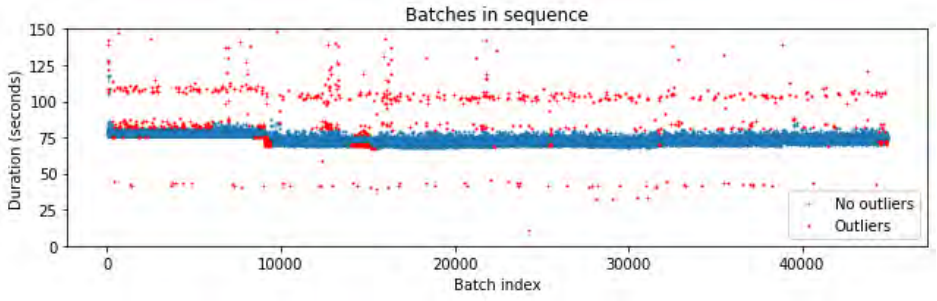


FIGURE 4.9: HA1_ZEEF durations sorted by start date, marked as outlier or not.

By applying the windowed IQR rule, 1,434 observations of the HA1_ZEEF durations (3.19%) are classified as an outlier, based on a selected window size of 1,000 batches. The markings of the HA1_ZEEF durations sorted on start date are shown in Figure 4.9. Around batch 10,000, the durations shift from approximately 78 seconds to 73 seconds, which results in approximately 250 wrongly detected outliers. In addition, around batch 15,000 many red dots are present in Figure 4.9, which are the result of very consistent sieve switching durations of 72 and 73 seconds (more than 50%). The consistent durations result in an IQR of 1 second, and therefore in a lower bound of 70.5 seconds and upper bound of 76 seconds.

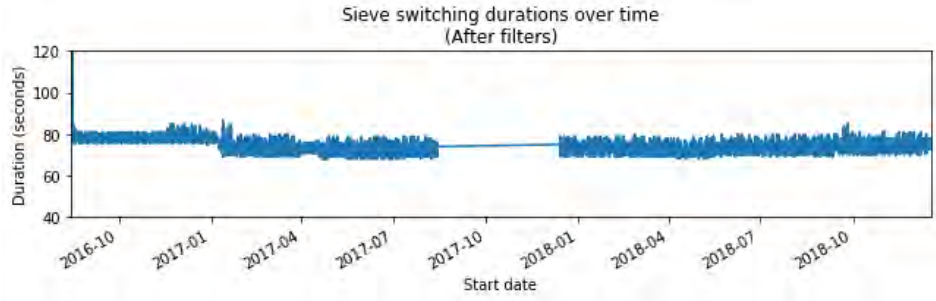


FIGURE 4.10: Remaining HA1_ZEEF durations plotted over time.

Removing the outliers results in the durations shown in Figure 4.10. Figure 4.10 shows also the need for a windowed-based outlier detection method because the median duration switches from approximately 78 to 73 seconds after the 1th of January in 2017. Note that there is a time gap of approximately 4 months between the first and second data dump (2017/08/15 – 2017/12/13). The change in the median value happens to be related to a replacement of an element of the sieve switching equipment.

Finally, the sieve size should not be changed after the sieve switching procedure started. However, this filter has no effect on the remaining batches. The final HA1_ZEEF durations range from 68 seconds until 127 seconds. The maximum duration of 127 occurs in the beginning, when not enough data was known to detect this maximum as outlier. After approximately 70 batches ranges the data from 68 until 87 seconds.

BU6_LOS

The third target variable is BU6_LOS, which represents the duration to empty the bunker under the hammer mills. This duration is measured for all 126,127 GML batches. Again, the windowed IQR rule is applied.

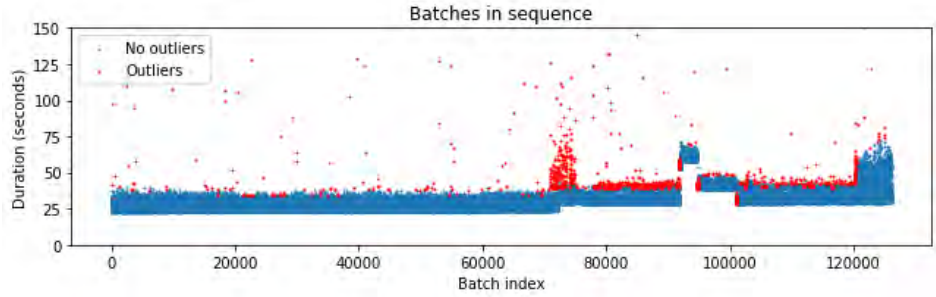


FIGURE 4.11: BU6_LOS durations sorted by start date, marked as outlier or not.

The window size that was selected for the windowed IQR rule is 500 batches, which was selected because it resulted in the least number of detected outliers. The results are shown in Figure 4.11, which shows the markings of the durations sorted by start date. Note that again a windowed outlier detection method was needed because the median value changes over time, which is probably due to the change in some machine setting that is not batch related.

By applying the windowed IQR rule, 1,805 batches (1.43%) were removed from the observations. Therefore, 124,322 observations are left, which range from 22 until 72 seconds, see Figure 4.12.

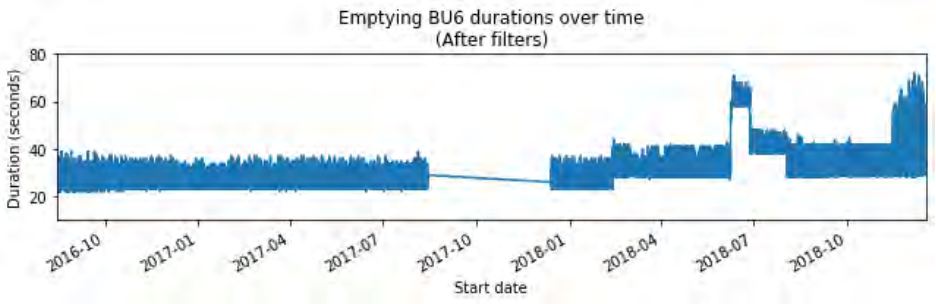


FIGURE 4.12: Remaining BU6_LOS durations plotted over time.

NM1_step

The fourth target variable is NM1_step, which represents the duration for mixing the ingredients in the mixer and adding liquids to it. This duration is measured for all 126,127 GML batches.

First of all, the windowed IQR rule was applied to remove the outliers from the data. The window size that was found to detect the least amount of outliers was the full history (126,127 batches). In this case, 9,100 batches (7.2%) were detected as outliers. See Figure 4.13 for the visualization of the classification of the windowed IQR rule with a window size of 126,127 batches.

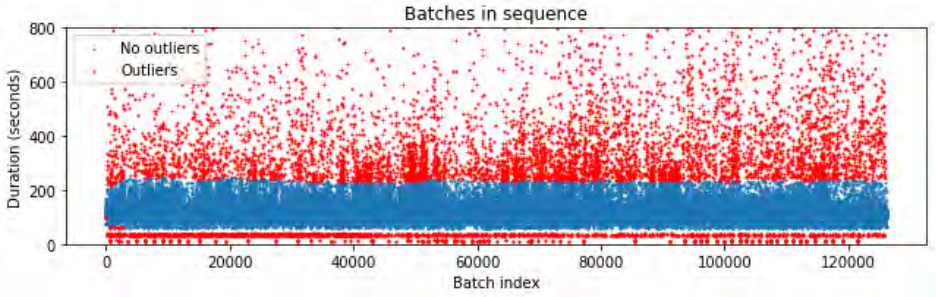


FIGURE 4.13: NM1_step durations sorted by start date, marked as outlier or not.

Secondly, all batches with a change in setpoints at the mixing unit after the production step has started should be removed. However, this filter has no effect because there are no changes made in the remaining batches after their start date.

Figure 4.14 shows a histogram of the final 117,027 NM1_step durations. The observations range from 60 until 251 seconds, with a mean value of 121.6 seconds.

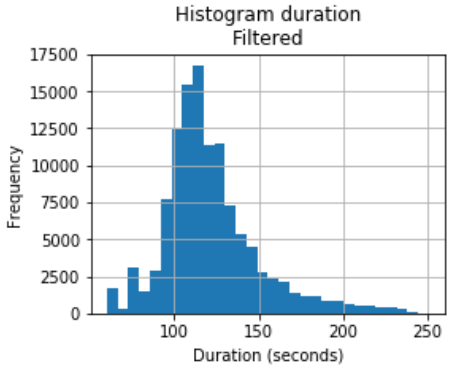


FIGURE 4.14: Histogram of the remaining NM1_step durations.

NM1_Los_step

The fifth target variable is NM1_Los_step, which represents the duration to empty the bunker under the mixing unit. Again, 126,127 observations are available of which the outliers need to be removed.

The selected value for the window size in the windowed IQR rule is 500 batches, which results in 463 outliers (0.37%). Figure 4.15 shows which mixing durations are detected as outliers and which not.

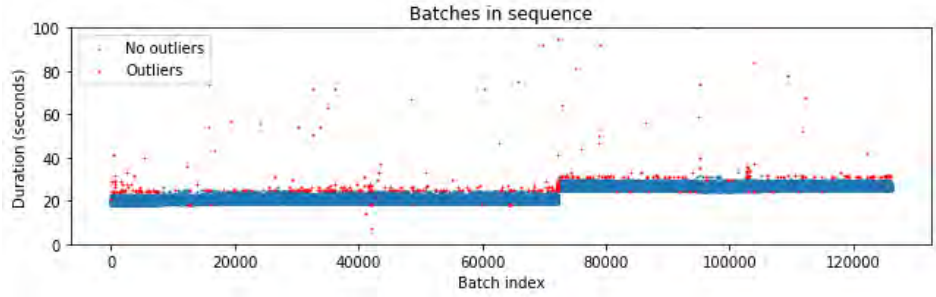


FIGURE 4.15: NM1_Los_step durations sorted by start date, marked as outlier or not.

The resulting 125,664 observations are shown in Figure 4.16. Again, the importance of a windowed outlier detection method becomes clear since a shift in the durations occurred on the 13th of February in 2018. Before this moment the mean emptying duration of the mixer was 20.4 seconds and after this moment 26.3 seconds. Probably a non-batch related machine setting was changed on that day, which negatively influenced the durations.

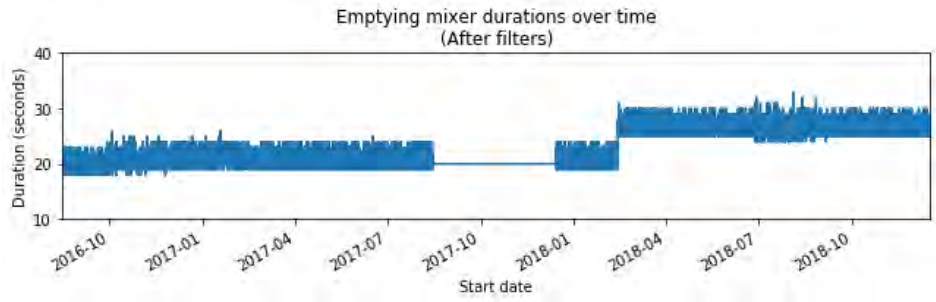


FIGURE 4.16: Remaining NM1_Los_step durations plotted over time.

MML_AFV_TR

The sixth target variable is MML_AFV_TR, which represents the duration to empty the bunker under the mixing unit. Since this transportation step includes a screw conveyor, the duration is strongly correlated with the batch size. This can also be observed in Figure 4.17, which shows a higher duration for bigger batch sizes.

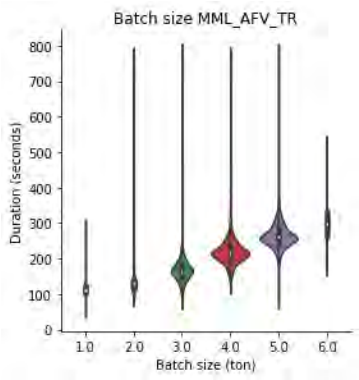


FIGURE 4.17: Violin chart which shows the correlation between the (produced) batch size and the MML_AFV_TR duration. Each violin shows the shape of the distribution and the total area of the violin indicates the number of data points.

Therefore, the windowed IQR rule is applied to the duration per 1,000 kg, instead of the total duration. The selected window size resulting in the smallest number of detected outliers is 1,000 batches, which detected 2,534 outliers (2.00%).

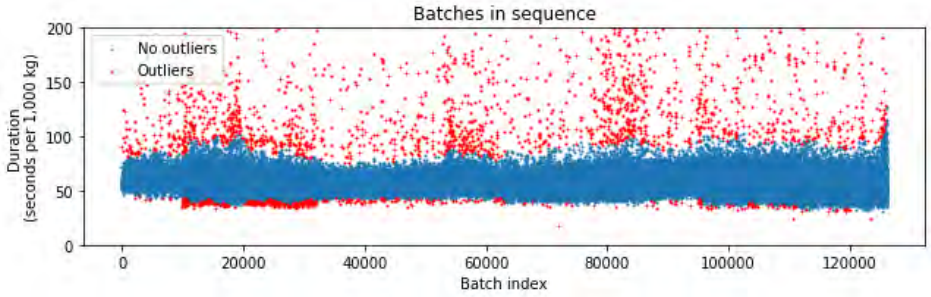
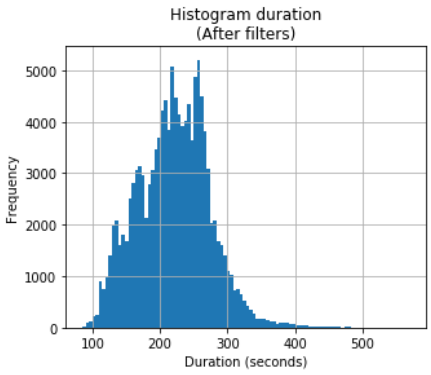


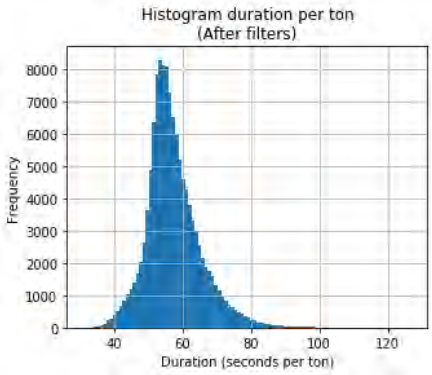
FIGURE 4.18: MML_AFV_TR durations sorted by start date, marked as outlier or not.

The second filter filters out all batches with a change in setpoints (the speed of the screw conveyor) after the bunker started emptying. This filter removes 79 batches.

Finally, 123,513 batches are left for the estimation of the MML_AFV_TR durations. The histograms of the MML_AFV_TR durations and the durations per 1,000 kg are shown in Figures 4.19A and 4.19B, respectively.



(A) Total production duration.



(B) Duration per 1,000 kg.

FIGURE 4.19: The distribution of the MML_AFV_TR durations after applying filters.

The remaining observations range from 85 seconds (1.42 minutes) until 570 seconds (9.5 minutes). The mean value is 220.46 seconds (3.67 minutes). The duration per 1,000 kg is found to be between 31.1 seconds and 126.8 seconds, with a mean value of 57.4 seconds.

MML_AFV_TR_NADRAAI

The last target variable of the GML is MML_AFV_TR_NADRAAI, which represents the duration to transport the final product of the GML to its destination: an end-silo or an input cell of a press line. As mentioned in the Preliminaries (Section 2), the transport duration can only be observed from the last batch of the same (special) article. Removing unusable batches results in 69,197 batches (54.86%). Since the MML_AFV_TR_NADRAAI durations strongly depend on the destination, the windowed IQR rule will be applied for each destination separately.

There are 13 different destinations observed in the dataset: 5 end-silos and 8 input cells for the press lines (two per press line). However, 3 end-silos occur only a few times: E-362 and E-384 occur 2 times and E-42 occurs 3 times.

The window size of the windowed IQR rule was also fitted for each destination separately, see Figures 4.20 and 4.21. This results in the window sizes and corresponding outlier percentages shown in Table 4.2. There are 67,492 remaining batches after filtering the outliers (2.46%).

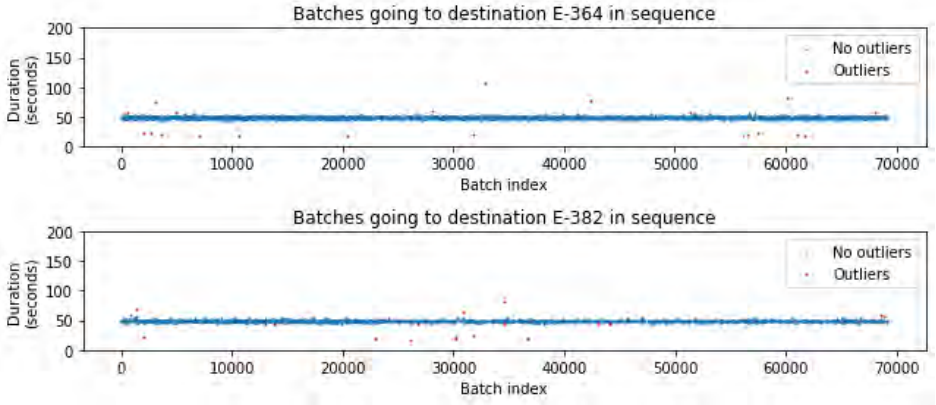


FIGURE 4.20: MML_AFV_TR_NADRAAI durations with an end-silo as destination sorted by start date, marked as outlier or not.

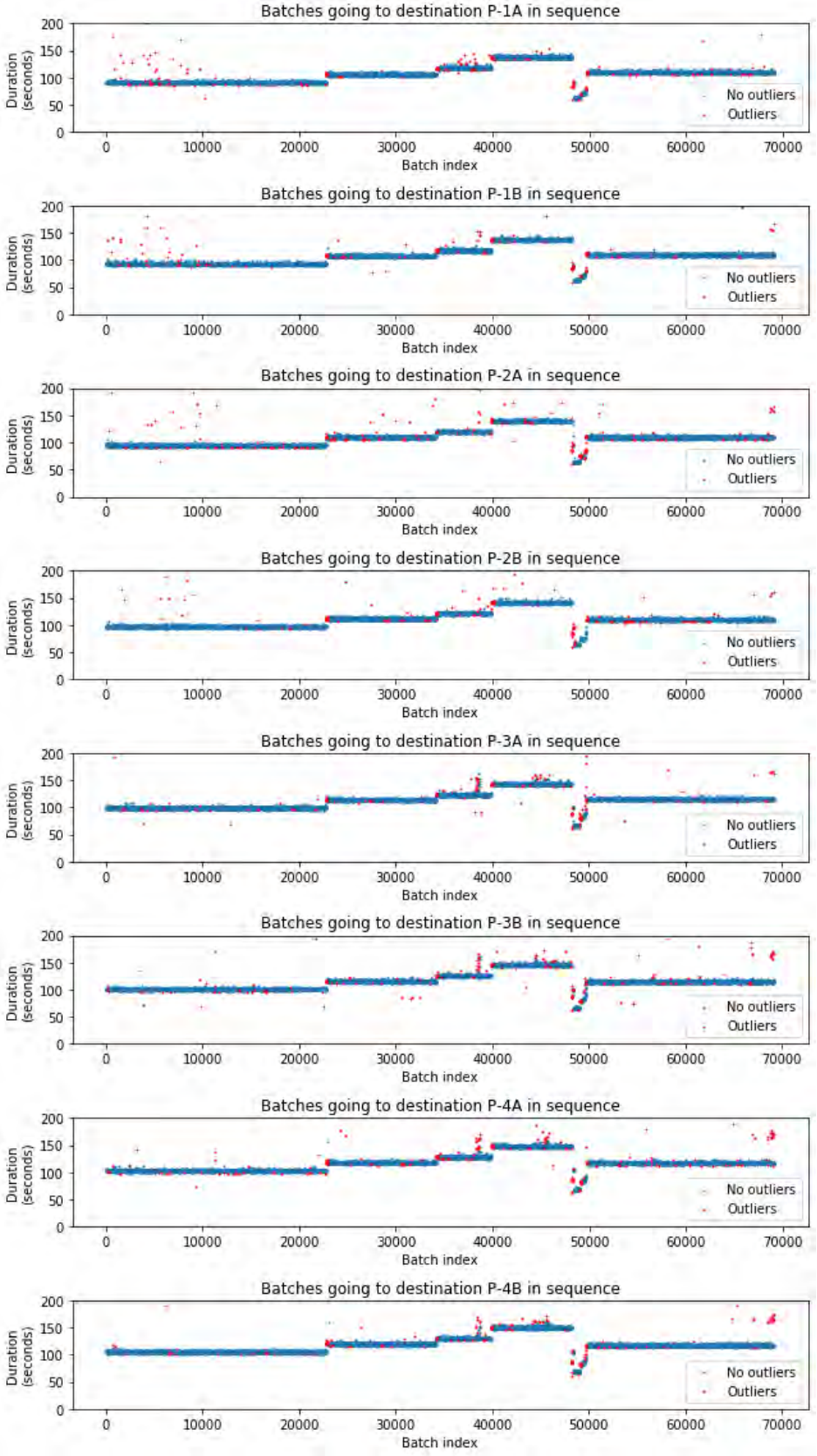


FIGURE 4.21: MML_AFV_TR_NADRAAI durations with an input cell of a press line as destination sorted by start date, marked as outlier or not.

Destination	Number of observations	Window size	Outliers percentage	Number of remaining observations
E-362	2	2	0%	2
E-364	2,298	500	0.96%	2,276
E-382	1,579	500	1.01%	1,563
E-384	2	2	0%	2
E-42	3	3	0%	3
P-1A	8,576	50	2.85%	8,332
P-1B	8,382	50	2.05%	8,210
P-2A	7,819	50	3.16%	7,572
P-2B	7,676	100	2.70%	7,469
P-3A	7,971	50	2.26%	7,791
P-3B	7,816	50	2.69%	7,606
P-4A	8,587	50	2.68%	8,357
P-4B	8,486	50	2.09%	8,309

TABLE 4.2: Results of the windowed IQR rule per destination.

4.3.2 Production durations at press lines

The next target variables take place at the press lines. The durations at the press units are handled differently from the previous target variables at the GML. The reason is that pressing is a continuous process. The pressing durations compose of a warm-up period (and cool-down period), further denoted as period A, and a period of maximal power of the press unit or supply, further denoted as period B, see Figure 4.22.

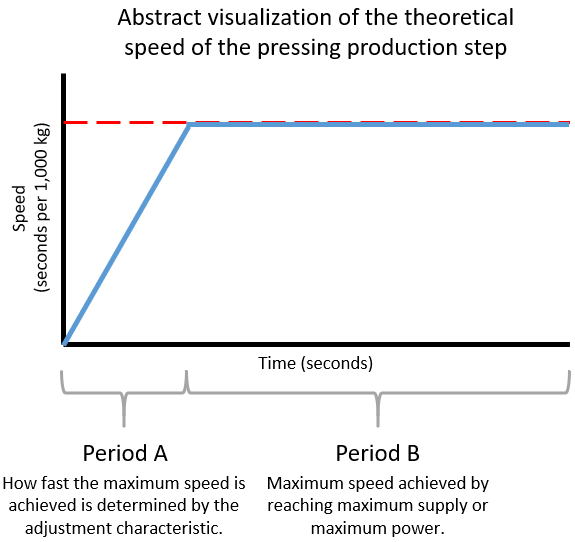


FIGURE 4.22: Abstract visualization of periods A and B.

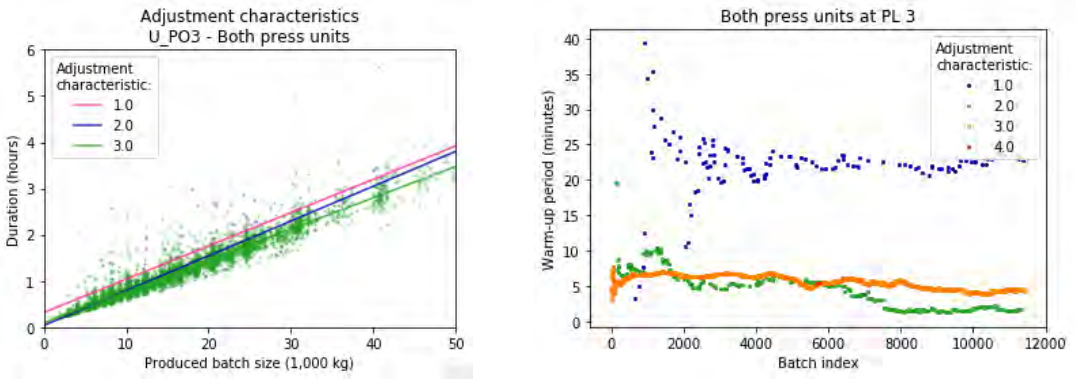
Interrupting the press line results in an extra warm-up period, and is therefore slower. This knowledge is used in the scheduling algorithm by splitting the total pressing duration in periods A and B. When a new schedule is made, the pressing duration starts with period A, followed by multiplying the batch size with the estimated speed in period B. Therefore, the estimation of the press unit target variables (PL_PO1, PL_PO2, PL_PO3 and PL_PO4) made in this research should contain both an estimation of period A (intercept) and an estimation of the speed in period B (slope).

How long the warm-up period takes is defined by the adjustment characteristic ('opregelkarakteristiek' in Dutch). One adjustment characteristic is composed of multiple settings, like the starting speed and the incremental speed that is added each number of seconds that is specified. Five possible adjustment characteristics can be used for the press line batches. The idea of an adjustment characteristic is that the operator could choose one of the characteristics instead of tuning all underlying settings.

By definition, the impact of an adjustment characteristic can change over time when the underlying settings are changed. However, the underlying settings are only available for the last three months of the dataset, and the adjustment characteristics themselves for the full dataset. Therefore, only the adjustment characteristics are used in this research.

To find the effect of an adjustment characteristic, a robust regression model is used. The model fits a robust regression line per adjustment characteristic to the pressing durations by using the batch sizes as a variable. The intercept of this regression line represents the warm-up period and the slope, the coefficient of the batch size, represents the speed during period B. To track differences in period A when the adjustment characteristic changes, the robust regression model is fitted per adjustment characteristic in a windowed fashion. More details of the robust regression model are given in Section 7.

In this research, the assumption is made that the adjustment characteristic determines the warm-up period. This assumption is important because it makes it possible to split periods A and B beforehand. Dropping this assumption would create the complexity of fitting both period A and the speed in period B without having the real value of periods A and B, but only the total pressing duration. By eliminating period A by assuming the intercept proposed by the robust regression model per adjustment characteristic is right, a machine learning model could be used relatively easily to predict the newly created target: the production duration minus period A divided by the batch size, which could be seen as the assumed 'real' speed in period B, see Table 4.3. Figure 4.23A shows (almost) parallel fitted robust regression lines for the different adjustment characteristics, which confirms the reasoning behind the assumption.



(A) Fitted robust regression per adjustment characteristic.

(B) Estimated values for the warm-up period with window size = 1,000 batches.

FIGURE 4.23: The adjustment characteristics of PL 3 durations using both press units.

The windowed robust regression method for determining period A of different adjustment characteristics was used for all press lines and each combination of used press units. So for PL 1 press unit A, press unit B and both press units, for PL 2 the single press unit, for PL 3 one and two press units (in parallel) and PL 4 press unit B and both press units. As mentioned, a windowed fashion is used. This is done by only looking at maximal B previous batches for fitting the robust regression line to the data of an adjustment characteristic. For B were the following values tested: 50, 100, 500, 1,000, 5,000, 10,000 and the size of the

dataset. The value 1,000 (or the total size if that is smaller than 1,000) was selected for each fit, which was chosen by visual comparison of the intercepts over time. Selecting less than 1,000 batches as window size results in discontinuity, which indicates uncertainty in the estimation of period A. Figure 4.23B shows the estimated warm-up periods of the batches at PL 3 which use both press units. The approximately horizontal lines show that the adjustment characteristics are not frequently changed for PL 3.

Target duration	**Period A variable**	**Speed period B variable (assuming period A is correct)**
PL_POi	PL_POi_intercept	PL_POi_slope $= \frac{\text{PL\_PO}i - \text{PL\_PO}i\text{\_intercept}}{\text{batch size (1,000 kg)}}$

TABLE 4.3: Separated target variables of the pressing duration of press line i, which is used for the estimation of the targets variables PL_PO1, PL_PO2, PL_PO3 and PL_PO4.

For the outlier removal, the same approach is used as at the GML. First, the outliers are found by applying the windowed IQR rule (Paragraph 4.2.3) to the batches, in which again the lower bound is defined by $1.5 * IQR$ and the upper bound by $3 * IQR$. In the case of the pressing durations, this method was applied to the assumed 'real' speeds of period B, which are shown in Table 4.3. Secondly, the batches were removed that contain changes in setpoints after the start of the production step.

PL_PO1

The eighth target variable of this research is PL_PO1, which represents the duration at the press units of PL 1. Previously was described how the estimation of period A, PL_PO1_intercept, was made. By assuming these estimations are correct, the assumed 'real' speed in period B, PL_PO1_slope, could be analysed. There are 15,308 batches available at PL 1: 312 batches using both press units, 1,634 using only press unit A and 13,362 using only press unit B.

The outliers are detected for PL_PO1_slope by applying the windowed IQR rule to the PL_PO1_slope values per press unit combination. This results in 102 outliers at press unit A (6.24%), 172 at press unit B (1.29%) and 15 at both press units (4.81%), by using each full history as the window size. See Figure 4.24 for a visualization of which PL_PO1_slope values are defined as outliers and which not.

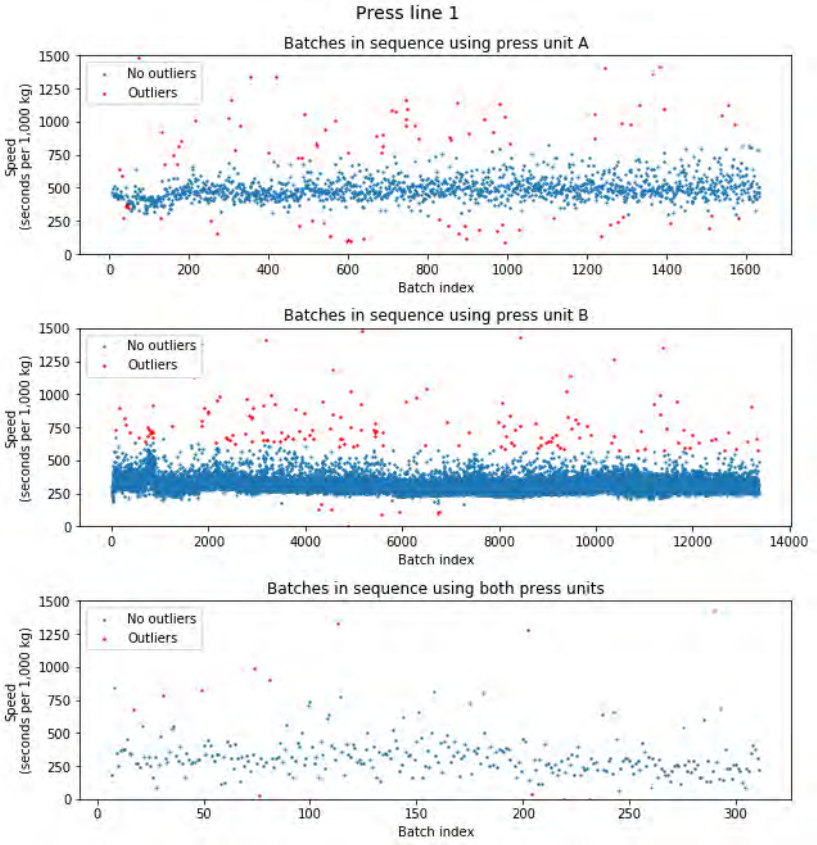


FIGURE 4.24: PL_PO1_slope values sorted by start date, marked as outlier or not.

After filtering the detected outliers, 15,019 batches are left (98.11%). From these remaining batches are the batches removed which contain changes to the setpoints while the press unit already started pressing, which are 409 batches (2.72%). Removing them results in 14,610 final batches. The histogram of the speed of the 14,610 final batches at PL 1 is shown in Figure 4.25.

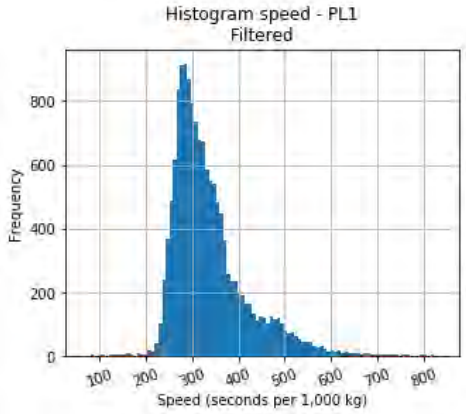


FIGURE 4.25: Final histogram of PL_PO1_slope.

PL_PO2

The ninth target variable is PL_PO2, which represents the duration at the press unit of PL 2. As before, this target is split into period A, PL_PO2_intercept, and in the speed in period B, PL_PO2_slope. PL_PO2_slope will be estimated by the models created in this research.

PL 2 contains only one press unit, at which 13,875 batches are produced. Detecting outliers results in 364 outliers (2.62%), with a window size of 500 batches. Figure 4.26 shows these outliers and the other values in a sequence.

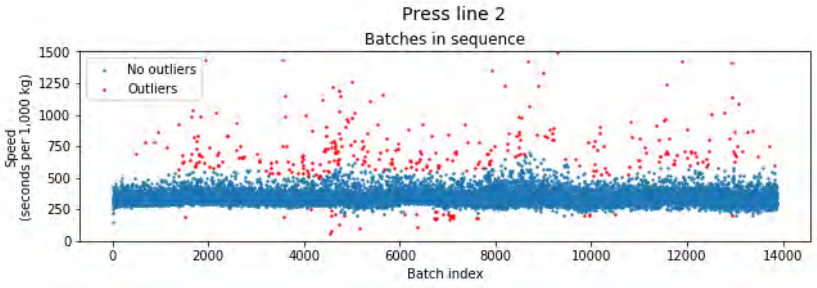


FIGURE 4.26: PL_PO2_slope values sorted by start date, marked as outlier or not.

Filtering the batches with changes after the press unit started pressing results in the removal of 455 batches. The final dataset for the estimation of the speed at press line 2 contains 13,056 batches. The histogram of the PL_PO2_slope values is given in Figure 4.27. It shows two peaks, which are further investigated in Section 6.

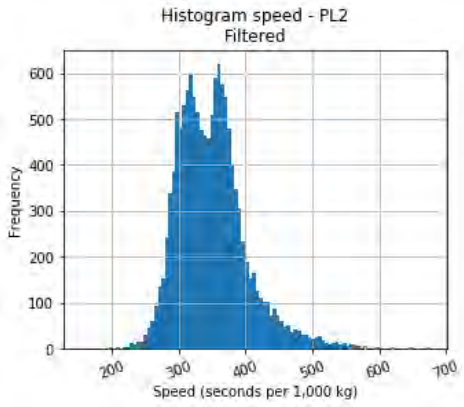


FIGURE 4.27: Final histogram of PL_PO2_slope.

PL_PO3

The tenth target variable is PL_PO3, which represents the duration at the press units of PL 3. Again, the same procedure is followed as for the other press lines.

There are 12,028 batches available for the estimation of PL_PO3_slope. 11,396 batches (94.75%) use two press units in parallel and 632 batches (5.25%) use only one press unit. By applying outlier detection, 389 batches were defined as an outlier by the use of two press units (window size = 5000), and 18 batches by the use of a single press unit (window size = full history). Figure 4.28 shows which values of PL_PO3_slope were defined as outliers and which not.

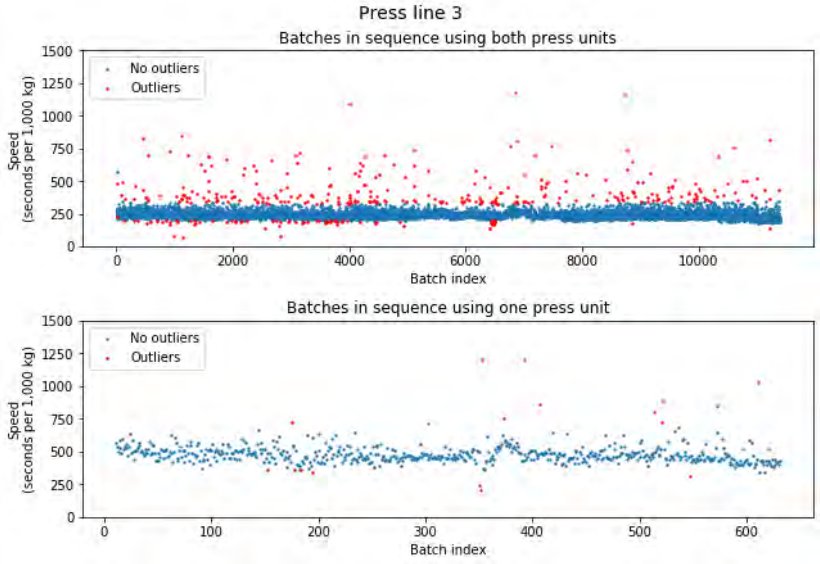


FIGURE 4.28: PL_PO3_slope values sorted by start date, marked as outlier or not.

Outlier removal results in 11,621 batches (96.6%). Besides, the batches with a change in setpoints after the start date are removed. The final 11,336 batches are used to estimate PL_PO3_slope, of which a histogram is shown in Figure 4.29.

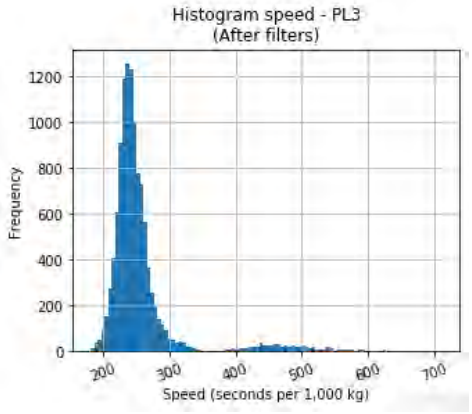


FIGURE 4.29: Final histogram of PL_PO3_slope.

Note that the speed while using two parallel press units is approximate twice the speed while using only one press unit. This explains the shape of Figure 4.29.

KO3_idle_time

The eleventh target variable is KO3_idle_time, which represents the cooling duration after the press units at PL 3. The KO3_idle_time durations can be observed for all PL 3 batches, which are 12,028 batches. Again, the windowed IQR rule is applied for the detection of outliers in the dataset. The results are shown in Figure 4.31, for a window size of 500. In contrast to the earlier selected window sizes was this selection not the one with the least number of detected outliers: 529 outliers with window size 500, and 527 outliers with a window size of the full history. The reason is that using a bigger window size results in a bigger IQR, because of the bigger variance in the cooling durations at the beginning of the dataset, which resulted in fewer detected outliers, see Figure 4.30. Since we want the outlier detection method to 'follow' the shape of the data over time, the window was set to 500,

which is the second window size with the fewest outliers. A window size of 500 results in a better visual fit, see Figure 4.31 compared to Figure 4.30.

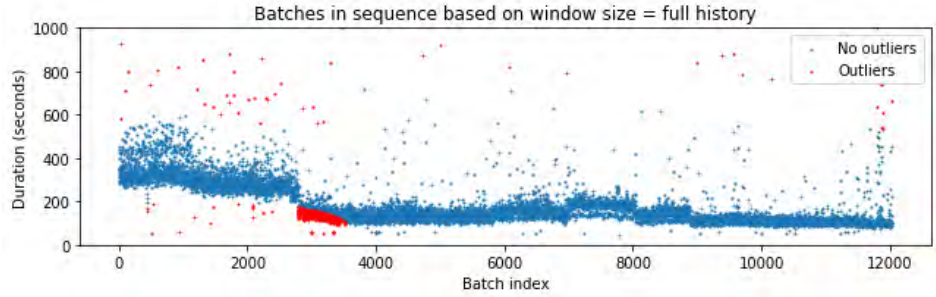


FIGURE 4.30: KO3_idle_time durations sorted by start date, marked as outlier or not, based on a window size of the full history.

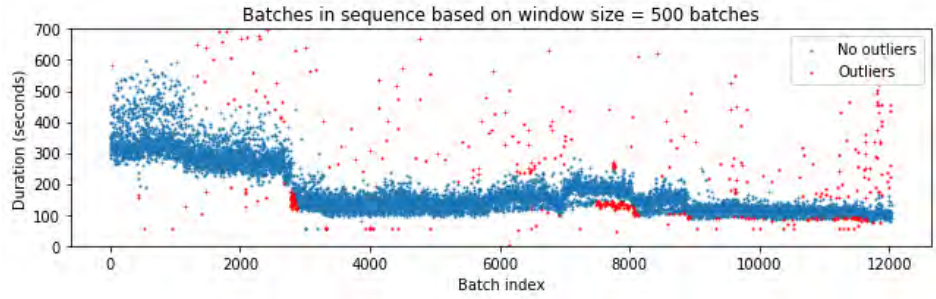


FIGURE 4.31: KO3_idle_time durations sorted by start date, marked as outlier or not, based on a window size of 500 batches.

Since there are no setpoints of the cooler available that could be changed after the cooling step started, the second filter does not affect. After removing the outliers, 11,499 batches are left, which are shown over time in Figure 4.32. The data ranges from 56 seconds to 597 seconds (9.95 minutes), with a mean value of 180 seconds (3 minutes).

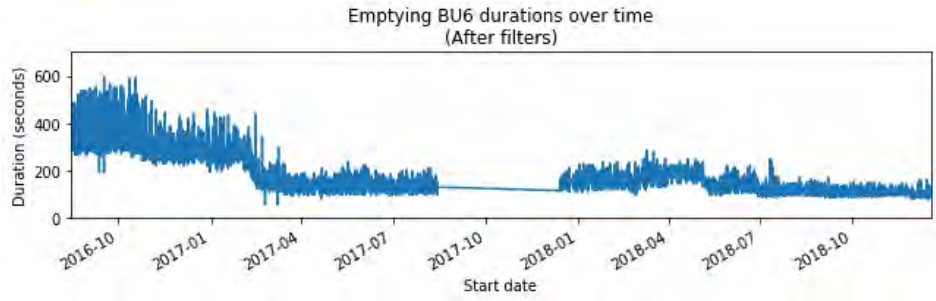


FIGURE 4.32: Remaining KO3_idle_time durations plotted over time.

PL_PO4

The last target variable is PL_PO4, which represents the duration at the press units of PL 4. This press line contains two press units: A and B, of which B is always used. Therefore, there are two possibilities: using press unit B, or both. Historically, 6,662 batches are using only press unit B, and 9,663 batches using both press units.

For the outlier detection of the batches using only press unit B is a window size of 1,000 used, and for the batches using both press units a window size of the full history. This

resulted in 160 outliers using press unit B (2.40%) and 184 outliers using both press units (1.90%). The batches are shown in sequence in Figure 4.33, in which the colour indicates if the PL_PO4_slope value is classified as an outlier or not.

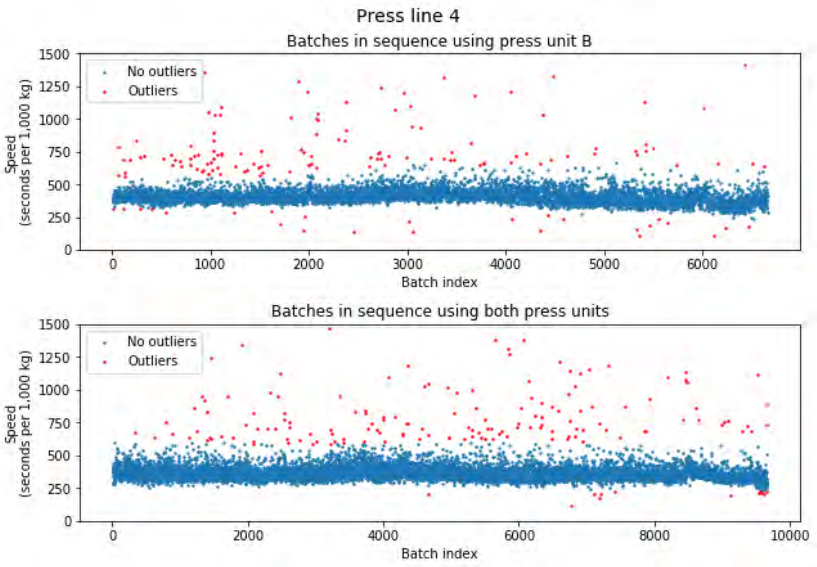


FIGURE 4.33: PL_PO4_slope values sorted by start date, marked as outlier or not.

The outliers are removed from the PL_PO4 observations, which results in 15,981 remaining batches (97.89%). Another 268 batches were removed because they contained changes after the production step was started.

The final dataset, which is used to estimate the PL_PO4 durations, contains 15,713 batches in total. The histogram of the assumed 'real' speed values is shown in Figure 4.34.

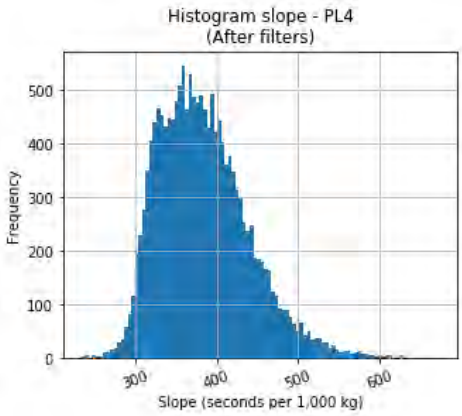


FIGURE 4.34: Final histogram of PL_PO4_slope.

5 Feature engineering

In the previous section, the collected dataset was described and the outliers were removed. The variables that will be used in the models, commonly called *features*, are described in this section. These features can be split into seven groups: batch size, time, setpoints, ingredients, weather, articles and storage unit related features.

5.1 Batch size

The first group that is used to estimate the production durations are the batch-size-related features, see Table 5.1. The production durations are assumed to be related to the produced amount, which results in the first feature *bookProducedKg*. Since the produced amount is not known in advance, but only the requested amount, it would make sense to use the requested amount as an indicator for the produced amount. However, the batch size of the GML is used as a global unit in the scheduling algorithm. Therefore, this research focuses on the produced amounts instead of the requested amounts in the estimation of the production durations. For future predictions, the produced amount could be estimated by, for example, multiplying the requested batch size at the GML with 1.025 for pellet products and with 1.01 for crumble products, as mentioned in Section 4.2.3. The produced amount at the press lines will then be the summation of the estimated produced amounts of the GML batches.

Feature name	**Database name**	**Description**
bookProducedKg	bookProducedKg	Produced batch size (in kg).
kgLiquids	deliveredAmountKg	Amount of dosed liquids at the mixer (in kg).

TABLE 5.1: Features based on the batch size.

In addition, the amount of dosed liquids is included as feature *kgLiquids*. The NM1_step durations consist of a mixing phase and a liquid dosing phase by definition. The amount of dosed liquids is assumed to be related to the liquid dosing duration.

5.2 Time

The second category of features contains time-based features. These features are created to comprehend the time, so the model could use them to find some trends in time. The features are shown in Table 5.2.

Feature name	Description	Value range
dayNr	Time since first observation in dataset, expressed in days.	Positive float
year	Year of observation.	Positive integer
month	Month of observation.	(1, ..., 12)
dayOfMonth	Day of month.	(1, ..., 31)
dayOfWeek	Weekday.	(Monday=0, ..., Sunday=6).
season	Season of the year.	(winter=1, spring=2, summer=3, autumn=4)

TABLE 5.2: Time features.

The first feature is *dayNr*, which indicates how many days past since the first observation of the production duration. This feature is introduced to capture information on when the production duration occurred in time. The *year* and *month* features could help the model to understand yearly and monthly patterns. The same holds for the features *dayOfMonth*, *dayOfWeek* and *season* for respectively inner monthly, inner weekly and seasonal patterns.

5.3 Setpoints

The setpoints are used to define the batch related settings of the machines. Since the machine settings directly influence the production durations, this information should be given to the model.

5.3.1 Hammer mill

The first setpoints are from the hammer mill (HA1). Table 5.3 lists the features and gives a description of them.

Feature name	Setpoint name (Dutch)	Description
HA1_supplyStartCapacity	HA1_TOEV Start capaciteit doseerrol	Start capacity of the supply conveyor of HA1 (in percentages).
HA1_supplyMaxCapacity	HA1_TOEV Max. capaciteit doseerrol	Maximum capacity of the supply conveyor of HA1 (in percentages).
HA1_supplyPower	HA1_TOEV Gewenst vermogen	Desired power of the supply conveyor of HA1 (in kW).
HA1_grindingSpeed	HA1_MAAL Gewenst toerental	Desired grinding speed of HA1 (in revolutions/minute).
HA1_sieveSize	HA1_ZEEF Gewenste zeefmaat	Desired sieve size of HA1 (in mm).
HA1_sieveSwitch	HA1_ZEEF Gewenste zeefmaat	Previous sieve size of HA1 (in mm) to current sieve size of HA1 (in mm).

TABLE 5.3: Features based on the setpoints of the hammer mill.

The desired sieve size (*HA1_sieveSize*) determines the maximum size of the materials after the grinding step. To grind the materials, the hammers need to rotate with a certain speed (*HA1_grindingSpeed*) for which the hammer mill needs power.

The supply conveyor, which doses the amount of material that goes into the hammer mill, determines the start of the HA1_TOEV duration. The supply conveyor starts with a certain start capacity (*HA1_supplyStartCapacity*), which is increased until the maximum capacity of the supply conveyor (*HA1_supplyMaxCapacity*) or the maximum power of the hammer mill (*HA1_powerSupply*) is reached. The supply conveyor stops when the material detector detects the hammer mill to be empty.

Finally, a feature *HA1_sieveSwitch* is created to keep track of the previous compared to the current sieve size. This is used to determine what kind of switch was made during the HA1_ZEEF durations.

5.3.2 Mixer

The NM1_step durations compose of the mixing duration and the liquid dosing duration. The mixing duration *NM1_totalMixDuration* can be created by the sum of the dry and wet mixing durations, which are defined in the setpoints of the batches produced at the mixer.

Feature name	Setpoint name (Dutch)	Description
NM1_totalMixDuration	NM1_Droge_Mengtijd + NM1_Natte_Mengtijd	Total mixing duration: sum of dry and wet mixing duration (in seconds).

TABLE 5.4: Features based on the setpoints of the mixer.

5.3.3 Screw conveyor

How long it takes to empty BU7, the bunker beneath the mixer depends on the speed of the screw conveyor. Therefore, a feature is created of this setpoint, see Table 5.5.

Feature name	Setpoint name (Dutch)	Description
BU7_speedScrewConveyor	BU7_Los Snelheid_schroef	Speed of screw conveyor (in percentage of maximum speed).

TABLE 5.5: Features based on the setpoints of the screw conveyor of BU7.

5.3.4 Press units

Since the press lines differ slightly from each other, not all setpoints exist on every press line. Therefore, the created features will be described per press line.

The setpoint related features that are present at PL 1 are shown in Table 5.6. This press line contains two press units in series. The features *PO1A_usePress* and *PO1B_usePress* indicate which press units are used to press the batch. Of these press units the desired power is given by *PO1A_pressPower* and *PO1B_pressPower*, respectively. Note that missing values are created for the features that are specific for a press unit if the press unit is not used for the production of the batch. When the materials are relatively 'difficult' to press, the power of the press is the bottleneck of this production step. However, when the materials are relatively 'easy' to press, the power of the press is not the bottleneck but the maximum capacity of the supply conveyor (*PO1_supplyCapacity*) is. It is not uncommon that during one batch the bottleneck shifts between the maximum power and the maximum supply.

Feature name	Setpoint name (Dutch)	Description
PO1_supplyCapacity	PO1_TOEV Capaciteit toevoer	Maximum capacity of the supply conveyor of PL 1 (in kg/second).
PO1_temperature	PO1_SDOS Gewenste temperatuur	Desired temperature of pressing (in Celsius).
PO1A_usePress	PO1_PO_1A Pers gebruiken	Use press unit A (Yes=1, No=0).
PO1B_usePress	PO1_PO_1B Pers gebruiken	Use press unit B (Yes=1, No=0).
PO1A_pressPower	PO1_PO_1A Gewenste vermogen pers	Desired power of press unit A (in kW).
PO1B_pressPower	PO1_PO_1B Gewenste vermogen pers	Desired power of press unit B (in kW).
PO1A_smearingOffset	PO1_PO_1A Versmeer offset pers	Offset of how much smearing is allowed at press unit A (in kW). If the offset is exceeded the supply is automatically reduced.
PO1B_smearingOffset	PO1_PO_1B Versmeer offset pers	Offset of how much smearing is allowed at press unit B (in kW). If the offset is exceeded the supply is automatically reduced.

TABLE 5.6: Features based on the setpoints of PL 1.

It is known that different variables have effect in the two scenarios. For example, the temperature (*PO1_temperature*) and the ingredient composition do have a direct impact on the pressing speed when the maximum power is reached because these variables contain information on how 'easy' it is to press the materials. However, when the maximum supply is reached, improving the circumstances of the press unit does not relate to faster pressing, because it is not the bottleneck.

Finally, 'smearing' could happen during pressing, which means that materials accumulate inside the pellet mill. See Figure 5.1 for a visualization of a pellet mill. When this happens, and the offset is exceeded (*PO1A_smearingOffset* or *PO1B_smearingOffset*), the supply is reduced. This kind of relationships should be recognized by the model proposed in this research.

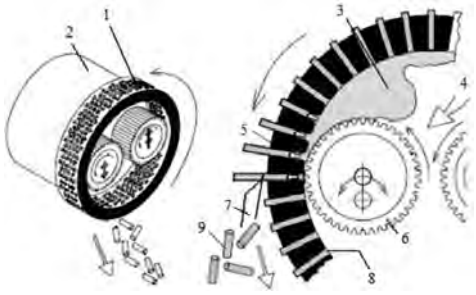


FIGURE 5.1: Ring die of pellet mill: 1-ring die; 2-gear; 3-crushed material; 4-material input; 5-die; 6-roller; 7-cutter; 8-pellet compressing hole; 9-pellet. [23], [24]

The second press line, PL 2, has only one press unit. Similar to PL 1, PL 2 has features for the maximum capacity (*PO2_supplyCapacity*), the temperature (*PO2_temperature*), the power of the press (*PO2_pressPower*) and the smearing offset (*PO2_smearingOffset*), see Table 5.7.

On top of that, PL 2 has a BOA, which could be used. *PO2_useBOA* determines if the BOA is used or not. When it is used, it starts with an opening of *PO2_BOAStartSize* millimetres, which is increased until the final opening size is reached. The final opening size has never been changed in plant X and is therefore not used as a feature. Since the BOA has a maximal power, *PO2_BOAPower*, the BOA could be another bottleneck for PL 2.

Feature name	**Setpoint name (Dutch)**	**Description**
PO2_supplyCapacity	PO2_TOEV Capaciteit toevoer	Maximum capacity of the supply conveyor of PL 2 (in kg/second).
PO2_temperature	PO2_SDOS Gewenste temperatuur	Desired temperature of pressing (in Celsius).
PO2_pressPower	PO2_PO Gewenste vermogen pers	Desired power of the press unit (in kW).
PO2_smearingOffset	PO2_PO Versmeer offset pers	Offset of how much smearing is allowed at the press unit (in kW). If the offset is exceeded the supply is automatically reduced.
PO2_rollingDistance	PO2_PO Rolafstand	Rolling distance of press unit (in mm).
PO2_useBOA	PO2_BOA Voorverdichten	Use BOA (Yes=1, No=0).
PO2_BOAStartSize	PO2_BOA Start opening spleet	Start opening of BOA (in mm).
PO2_BOAPower	PO2_BOA Gewenst vermogen BOA	Desired power of BOA (in kW).

TABLE 5.7: Features based on the setpoints of PL 2.

The last difference between PL 2 and PL 1 is the feature *PO2_rollingDistance*. This feature defines the distance between the rollers and the die of the pellet mill, see Figure 5.1. For PL 1, this setpoint is not optional, but for PL 2 it is.

PL 3 is different from the other press lines; it has two parallel press units. This results in two different setpoints for the temperature (*PO3A_temperature* and *PO3B_temperature*) and supply capacity (*PO3A_supplyCapacity* and *PO3B_supplyCapacity*), see Table 5.8. Normally both press units are used simultaneously. However, it is possible that one of them is not available, for example, because it is broken, which is then indicated by the features *PO3A_usePress* and *PO3B_usePress*.

Feature name	Setpoint name (Dutch)	Description
PO3A_usePress	PB3_VUL Pers A gebruiken	Use press unit A (Yes=1, No=0).
PO3B_usePress	PB3_VUL Pers B gebruiken	Use press unit B (Yes=1, No=0).
PO3A_supplyCapacity	PO3A_TOEV Capaciteit toevoer	Maximum capacity of the supply conveyor of PL 3 to press unit A (in kg/second).
PO3B_supplyCapacity	PO3B_TOEV Capaciteit toevoer	Maximum capacity of the supply conveyor of PL 3 to press unit B (in kg/second).
PO3A_temperature	PO3A_SDOS Gewenste temperatuur	Desired temperature of pressing at press unit A (in Celsius).
PO3B_temperature	PO3B_SDOS Gewenste temperatuur	Desired temperature of pressing at press unit B (in Celsius).
PO3A_pressPower	PO3_PO_3A Gewenste vermogen pers	Desired power of press unit A (in kW).
PO3B_pressPower	PO3_PO_3B Gewenste vermogen pers	Desired power of press unit B (in kW).
PO3A_smearingOffset	PO3_PO_3A Versmeer offset pers	Offset of how much smearing is allowed at press unit A (in kW). If the offset is exceeded the supply is automatically reduced.
PO3B_smearingOffset	PO3_PO_3B Versmeer offset pers	Offset of how much smearing is allowed at press unit B (in kW). If the offset is exceeded the supply is automatically reduced.

TABLE 5.8: Features based on the setpoints of PL 3.

Similar to the other press lines, PL 3 has features for the power of the press units (*PO3A_pressPower* and *PO3B_pressPower*) and the smearing offset (*PO3A_smearingOffset* and *PO3B_smearingOffset*). These features and the features for the maximal capacity represent the bottleneck of PL 3.

Finally, the last press line, PL 4, is very similar to PL 1. However, the second press unit, press unit B, could not be skipped. Therefore, only the feature *PO4A_usePress* is needed to describe which press units are used, instead of two features. When the feature *PO4A_usePress* is set to 'no', the batch is only produced at press unit B, otherwise, both press units are used. The other features of PL 4 are similar to the features of PL 1, see Table 5.9.

Feature name	Setpoint name (Dutch)	Description
PO4_supplyCapacity	PO4_TOEV Capaciteit toevoer	Maximum capacity of the supply conveyor of PL 4 (in kg/second).
PO4_temperature	PO4_SDOS Gewenste temperatuur	Desired temperature of pressing (in Celsius).
PO4A_usePress	PO4_PO_4A Pers gebruiken	Use press unit A (Yes=1, No=0).
PO4A_pressPower	PO4_PO_4A Gewenste vermogen pers	Desired power of press unit A (in kW).
PO4B_pressPower	PO4_PO_4B Gewenste vermogen pers	Desired power of press unit B (in kW).
PO4A_smearingOffset	PO4_PO_4A Versmeer offset pers	Offset of how much smearing is allowed at press unit A (in kW). If the offset is exceeded the supply is automatically reduced.
PO4B_smearingOffset	PO4_PO_4B Versmeer offset pers	Offset of how much smearing is allowed at press unit B (in kW). If the offset is exceeded the supply is automatically reduced.

TABLE 5.9: Features based on the setpoints of PL 4.

5.3.5 Crumbler

For the production of crumbled feed is a crumbler used, which crumbles pellets. The use of the crumbler is likely to affect the moment of emptying the cooler because it indicates the next destination in the continuous process. Availability of the next destination in a continuous process affects the duration. Therefore, a feature is created based on the corresponding setpoint of the crumbler, see Table 5.10.

Feature name	Setpoint name (Dutch)	Description
PO3_crumble	OB3_KRUIM Kruimelen	Crumble product (Yes=1, No=0).

TABLE 5.10: Features based on the setpoints of the crumbler at PL 3.

5.4 Ingredients

As already mentioned in the introduction, we expect the ingredient composition of a batch to affect the production durations. In order to select the ingredients that are contained in the batch, the ingredients are selected based on their input location. By looking at the input location, four ingredient location groups are created, see Table 5.11.

Ingredient group	Percentage features	nIngredients feature
Until hammer mills	84 features representing the ingredient percentages of a hammer mill batch.	Number of ingredients of hammer mill batch.
Liquids at mixer	16 features representing the ingredient percentages of dosed liquids at the mixer.	Number of different liquids at mixer added to the batch.
Until mixer	174 features representing the ingredient percentages of a BU7 batch after the mixer.	Number of ingredients of BU7 batch.
All	176 features representing the ingredient percentages of a press line batch, which contains all ingredients of the whole production process.	Number of ingredients of press line batch.

TABLE 5.11: Percentages and nIngredients features per ingredient group.

First of all, the ingredients of the hammer mills are obtained by the ingredients of the dosing steps above the hammer mills. This results in 84 different ingredients. For each of these ingredients, a feature *articleId* is created which contains the percentage of the ingredient amount compared to the total batch amount, see Table 5.11. The percentage instead of the amount is used, to make the batches comparable with each other. In addition, another feature, *nIngredients*, is created which represents the number of ingredients that are present in the batch, see Table 5.11.

The second ingredient location group contains all liquids that are added at the mixer. This group does not include the previous ingredient location group, because only the liquids are relevant for the NM1_step durations, and not the ingredients that were already present in the batch. For this ingredient location group, the same features are created as for the first ingredient location group, see Table 5.11.

The third ingredient location group represents all the ingredients that could be present in a batch at the bunker below the mixer (BU7). This group contains, therefore, all ingredients that are present at the hammer mills (84), as both the liquids (16) and non-liquids (74) that are added at the mixer. This results in 174 ingredient percentage features. In addition, the feature *nIngredients* is added which counts the number of ingredients that are present in the batch.

Finally, the last ingredient location group contains all ingredients that are added somewhere in the process to the batch. Therefore, the last ingredient location group contains all the ingredients that were present at BU7. On top of that, it includes the (sticky) ingredients that are added during the transportation to the next destination after the production at the GML. Since there are only two unique ingredients added during the transportation step, this group contains only two more ingredients than the third ingredient location group, see Table 5.11. Again, for each ingredient is a feature created, and another feature to count the number of ingredients in the batch, see Table 5.11.

5.4.1 Ingredient Groups

To limit the number of ingredient features per ingredient location group, we can use a summary by grouping all ingredients by their ingredient article group. This results in a feature per ingredient article group representing the percentage of which the batch consists of the

ingredient article group. For example, to obtain the percentage of liquids, all liquid ingredients in the batch are summed together. In total, there are five ingredient article groups, which results in five features, see Table 5.12. These summarizing features are created for the ingredients of the four ingredient location groups of Table 5.11.

Feature name	**Ingredient group**	**Description**
igrGroup_ENK	ENK	Percentage of ingredients amount in batch belonging to single feeds.
igrGroup_GR	GR	Percentage of ingredients amount in batch belonging to raw materials.
igrGroup_PM	PM	Percentage of ingredients amount in batch belonging to premixes.
igrGroup_UNKNOWN	None	Percentage of ingredients amount in batch of which the ingredient group is missing.
igrGroup_VL	VL	Percentage of ingredients amount in batch belonging to liquids.

TABLE 5.12: Features based on the ingredient group percentages.

5.5 Weather

The data collection process of the KNMI weather dataset was described in Section 4.1. Many weather variables are available, but only the variables of which some potential influence on the production durations is expected are included in this research. This results in the following list of variables, see Table 5.13.

Feature name	Variable name KNMI	Description
windSpeedMean	FHVEC	Vector mean windspeed (in 0.1 m/s).
windDirectionMean	DDVEC	Vector mean wind direction in degrees (360=north, 90=east, 180=south, 270=west, 0=calm/variable).
temperatureMean	TG	Daily mean temperature (in 0.1 degrees Celsius).
temperatureMin	TN	Minimum temperature (in 0.1 degrees Celsius).
temperatureMax	TX	Maximum temperature (in 0.1 degrees Celsius).
precipitationDuration	DR	Precipitation duration (in 0.1 hour).
precipitationAmount	RH	Daily precipitation amount (in 0.1 mm) (-1 for <0.05 mm).
relAtmosphericHumidityMean	UG	Daily mean relative atmospheric humidity (in percentages).

TABLE 5.13: Features based on the KNMI weather data.

The first two weather features, *windSpeedMean* and *windDirectionMean*, contain information about the wind. The *windDirectionMean* is created by taking the average over the wind direction in degrees (360=north, 90=east, 180=south, 270=west, 0=calm/variable). By the combination of the wind speed and the wind direction, the effect of the wind is examined. In addition, three temperature features, *temperatureMean*, *temperatureMin* and *temperatureMax*, representing the daily mean, minimum and maximum temperature respectively, are used to investigate the effect of the temperature on the production durations. The precipitation and humidity are evaluated too. This results in three other features: the *precipitationAmount* and *precipitationDuration*, and the *relAtmosphericHumidityMean*, see Table 5.13.

5.6 Article

The next group of features provides information of the previous observations of the same article. Since there are 959 different articles, creating one feature for each would lead to many different features and associated complexity. To help the model, features were created by aggregating information of the article, see Table 5.14. These features have as purpose to indicate how fast the previous batches of the same article were.

Feature name	Aggregation function	Aggregated group
articleMedian5	Median duration	Previous 5 batches of the same article id.
articleMin5	Minimum duration	Previous 5 batches of the same article id.
articleMax5	Maximum duration	Previous 5 batches of the same article id.
articleGroupMedian5	Median duration	Previous 5 batches of the same article group.
articleGroupMin5	Minimum duration	Previous 5 batches of the same article group.
articleGroupMax5	Maximum duration	Previous 5 batches of the same article group.
articleRecipeGroupMedian5	Median duration	Previous 5 batches of the same article recipe group.
articleRecipeGroupMin5	Minimum duration	Previous 5 batches of the same article recipe group.
articleRecipeGroupMax5	Maximum duration	Previous 5 batches of the same article recipe group.

TABLE 5.14: Features based on aggregated article information.

The first three features are created by taking the median, minimum and maximum value of the duration of the last 5 batches of the same article (*articleMedian5*, *articleMin5* and *articleMax5* respectively). The selection of 5 batches was made in order to coop with both noise and time effect. The same is done for article group (*articleGroupMedian5*, *articleGroupMin5* and *articleGroupMax5*) and recipe group (*articleRecipeGroupMedian5*, *articleRecipeGroupMin5* and *articleRecipeGroupMax5*).

The information that is aggregated depends on the target variable. If the target variable shows a strong correlation with the batch size, the duration per 1,000 kg is used, otherwise the total duration. In case of the press durations, the target speed values are used.

Finally, another feature is created, that specifies the specific weight of the article (*articleSpecificWeight*), see Table 5.15.

Feature name	Description
articleSpecificWeight	Specific weight of produced article (in kg/m^3).

TABLE 5.15: Features based on the article.

5.7 Storage unit

For the duration of the transportation system, the destination is important. Therefore, the last included feature *toStorageUnitNameLast* indicates which storage unit is the last destination of the batch, see Table 5.16.

Feature name	Setpoint name (Dutch)	Description
toStorageUnitNameLast	toStorageUnitName	The last destination of the transported batch.

TABLE 5.16: Features based on the destination of the transportation system after the GML.

6 Feature analysis

The features that are created for the estimation of the production durations were described in the previous section. These features will be analysed in this section in the context of each target variable.

The Pearson correlation coefficient is used as a correlation measure. Both the correlation with the target variable and the correlation between features is measured with this coefficient. The Pearson correlation indicates how much linear correlation there is between two variables. The Pearson correlation $\rho_{X,Y}$ can be calculated as follows, see Equation (6.1). The $cov(X, Y)$ denotes the covariance between variables X and Y, and σ_X and σ_Y are respectively the standard deviation of variable X and Y.

$$\rho_{X,Y} = \frac{cov(X, Y)}{\sigma_X \sigma_Y} \text{ with } \rho_{X,Y} \in [0, 1] \tag{6.1}$$

When $\rho_{X,Y}$ is close to 1, it indicates that X and Y are positively correlated and a value close to -1 indicates a negative correlation. A value close to zero indicates that there is no linear correlation between the two variables. Note that a strong correlation is not the same as a causal relation.

6.1 HA1_TOEV

The first production duration is the duration of the grinding step at the GML: HA1_TOEV. In total there are 120,527 observations available for this target. In Figure 6.1 is shown how these observations are spread over the week. The production at plant X is much higher during the working week than at the weekend. In addition, a small increase in the number of batches per day can be observed during the working week.

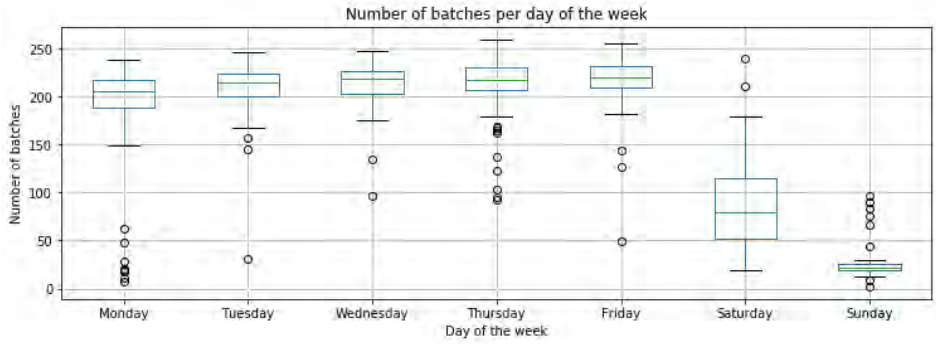
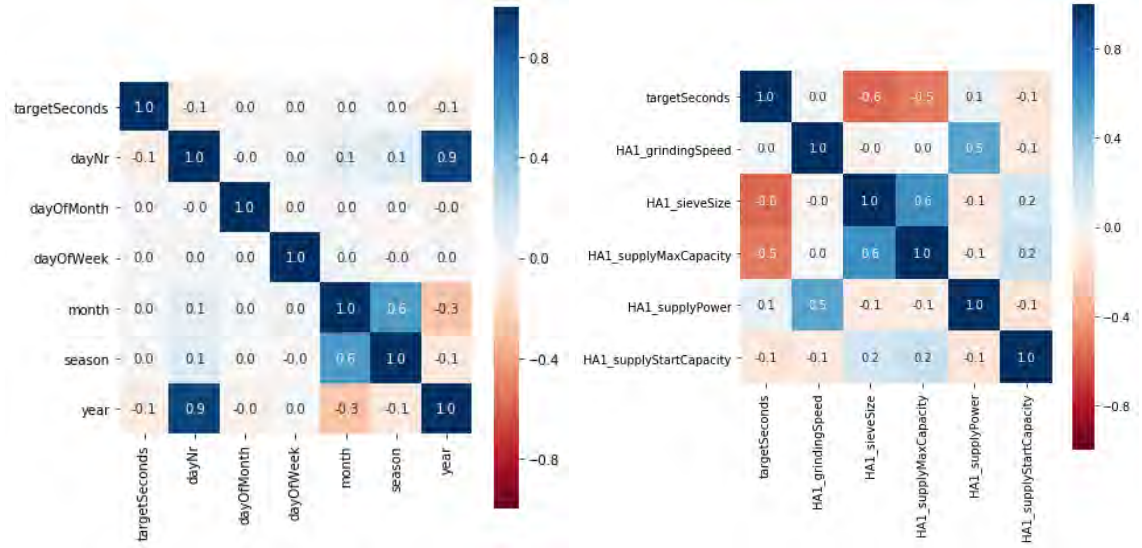


FIGURE 6.1: Number of HA1_TOEV observations per day of the week.

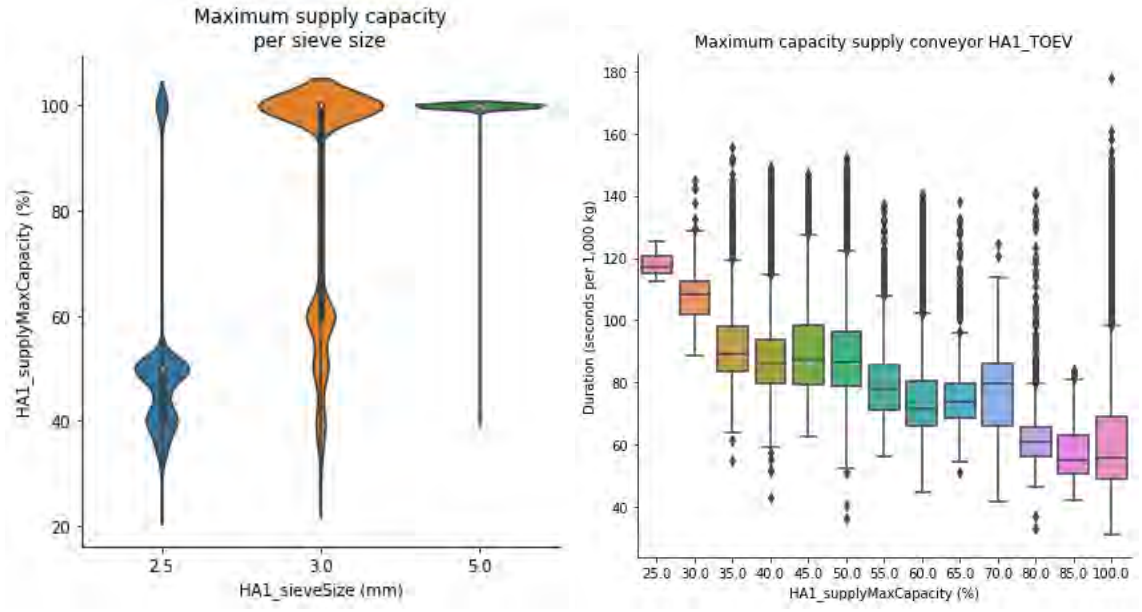
In the previous section, many features were created. The correlation between HA1_TOEV durations and the batch size was already mentioned in Section 4.3 (Figure 4.6A), which has a Pearson correlation of 0.6.

The second group of features are the time features, which show little correlation with the HA1_TOEV durations, see Figure 6.2A. Only the Pearson correlations between HA1_TOEV and the day number and year are not zero but are -0.1. In contrast, there is a strong correlation between some time features. For example, the month number and the season have a Pearson correlation of 0.6, and the day number has a Pearson correlation of 0.9 with the year number.



(A) Time features. (B) Setpoint features.

FIGURE 6.2: Pearson correlation between features and HA1_TOEV durations (targetSeconds).



(A) Sieve size. (B) HA1_TOEV duration per 1,000 kg.

FIGURE 6.3: Maximum supply capacity.

The third group of features are the setpoint features. In Section 4.3 was already mentioned that the HA1_TOEV durations and the sieve size show much correlation (Figure 4.6B). The same can be concluded from the Pearson correlation, which is -0.6, see Figure 6.2B. Besides the sieve size, the maximum supply capacity shows a high correlation with

the HA1_TOEV duration too, see Figure 6.3B, which has a Pearson correlation of -0.5. The features themselves are correlated as well (Pearson correlation = 0.6), see Figure 6.3A. When a sieve size of 2.5 mm is used, the maximum supply capacity is in 90% of the batches smaller than 60%. When the sieve size is 3 mm, in approximately 65% of the batches is the maximum supply capacity bigger than 80%. Finally, if a sieve size of 5 mm is used, the maximum supply capacity is smaller than 100% in only less than 5% of the batches.

The fourth group of features are the ingredient features. The Pearson correlations of the ingredient percentage features are shown in Figure A.1 in the Appendix. The correlations are computed against the HA1_TOEV durations per 1,000 kg in order to account for the dependency on the batch size. Roughly speaking, the ingredients *5333*, *975*, *993* and *6552* result in smaller HA1_TOEV durations and *1074*, *964*, *970* and *974* in a larger HA1_TOEV duration. Note that this only shows the correlation. This does not necessarily have to be the cause of the production duration. For example, it may be that one specific ingredient is used more for "fine" articles (small sieve size) and therefore correlate with a longer production time.

The number of ingredients is slightly negatively correlated (Pearson correlation = -0.2) with the production duration per 1,000 kg. An explanation could be that the ingredients which are more difficult to grind, appear more often in a smaller composition (fewer ingredients) than the ingredients that are easier to grind.

The ingredients that are ground at the hammer mill could be divided into two ingredient groups: *GR* (raw material) and *PM* (premix). The raw materials are slightly positively correlated with the production duration per 1,000 kg (Pearson correlation = 0.1), which means the more raw materials, the slower grinding, and premixes are slightly negatively correlated with the production duration per 1,000 kg (Pearson correlation = -0.1), which means the more premixes, the faster grinding.

The fifth group of features are the weather features. However, none of these features shows a linear correlation (Pearson correlation = 0) with the HA1_TOEV durations, see Figure A.2 in the Appendix.

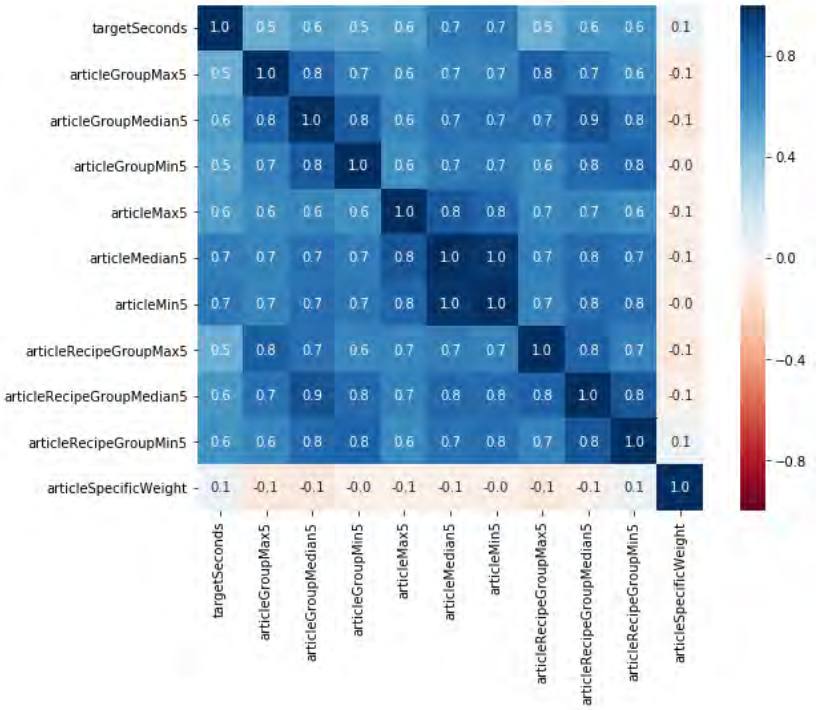
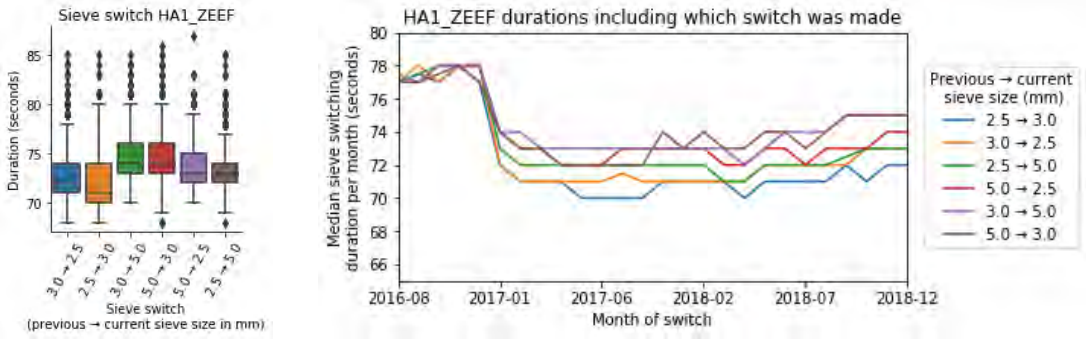


FIGURE 6.4: Pearson correlation between article features and HA1_TOEV durations (targetSeconds).

Finally, the last group of features are the article features. There is a very high correlation (Pearson correlation = 0.7) between the median duration of the previous five batches of the same article and the current batch, see Figure 6.4.

6.2 HA1_ZEEF

The second target variable is HA1_ZEEF: the duration of switching the sieve of the hammer mill. In Section 4.3, the HA1_ZEEF durations were already shown over time (Figure 4.10). On the first of January 2017, the sieve switching duration was significantly reduced.



(A) Boxplots of HA1_ZEEF durations per sieve switch.

(B) HA1_ZEEF durations over time, including which sieve switch was performed.

FIGURE 6.5: Relation sieve switch and HA1_ZEEF durations.

The HA1_ZEEF durations can be split to which sieve switch was performed. This sieve switch feature is linearly correlated with the HA1_ZEEF durations (Pearson correlation = 0.2). A relation can also be observed from Figures 6.5A and 6.5B. The switch from 2.5 mm to

3 mm and reversed have a smaller duration than the switch from 2.5 mm to 5 mm and reversed. To test if these durations are significantly different over the performed sieve switches, a two-sample t-test can be performed. This test can be applied to two samples and tests if the mean is significantly different. When a p-value is observed that is smaller than a certain α, the hypothesis that the means are equal is rejected, which means that there is a significant difference in the means of the two sample sets. Common values of α are 0.01 (1%), 0.025 (2.5%) or 0.05 (5%). When the t-test is applied to the HA1_ZEEF durations of every combination of the sieve switches, the durations are all significantly different (p-value « 0.01), except for the sieve switches 3 to 5 mm and 5 to 3 mm which has a p-value of 0.047, which is not rejected for an α of 0.01. We can conclude that the HA1_ZEEF durations are significantly different over the sieve switches (only 3 to 5 and 5 to 3 mm can be considered as the same), which indicates the importance of including the sieve switch in the proposed model.

Note that the spread in the HA1_ZEEF durations per sieve switch is very small. In addition, there was no correlation found between the HA1_ZEEF durations and the other features. Therefore, we conclude that this production duration is not the focus of this research. The proposed model will be a simple time-based model that calculates the median duration of the last B batches with the same sieve switch.

6.3 BU6_LOS

The third target variable is BU6_LOS: the duration to empty the bunker below the hammer mills. In Section 4.3, the BU6_LOS durations were already shown over time (Figure 4.12). The BU6_LOS duration shifts several times over the past two years. This results in a correlation between the time features and the BU6_LOS durations, see Figure 6.6.

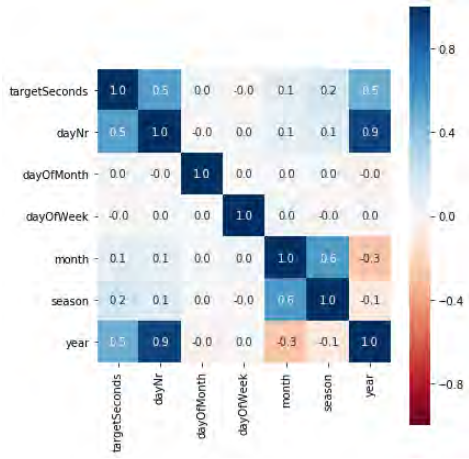


FIGURE 6.6: Pearson correlation between time features and BU6_LOS durations (targetSeconds).

From this analysis can be concluded that the BU6_LOS durations hardly show any correlation with non-time features. Therefore, we conclude that a simple time-based model is most appropriate for the estimation of the BU6_LOS durations. The proposed model will be a simple time-based model that calculates the median duration of the last B batches.

6.4 NM1_step

The fourth target variable is NM1_step: the duration to mix the ground materials and the liquids in the mixer. The liquid amount has a large correlation (Pearson correlation = 0.5) with the NM1_step durations compared to the other features suggested in the previous section (Section 5). This relation can also be observed in Figure 6.8A. The batch size is linearly correlated with the liquid amount (Pearson correlation = 0.4) as well, but it is less correlated (Pearson correlation = 0.1) with the NM1_step durations, see Figure 6.7.

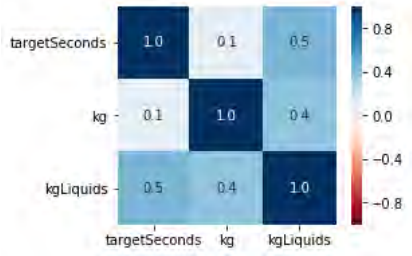
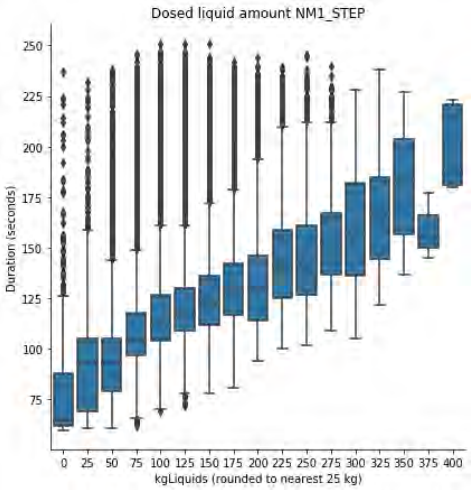
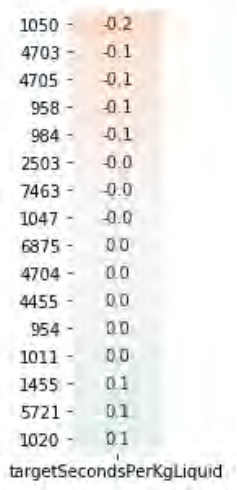


FIGURE 6.7: Pearson correlations of batch size features and NM1_step durations.

When looking at the specific liquids that are added at the mixer, we can calculate the seconds per liquid kg by subtracting the median duration without liquids (64 seconds) and divide it by the liquid amount. In this way, the liquid dosing duration can be compared with which liquids are dosed. Since multiple liquids are dosed simultaneously, the correlation gives only an indication of which liquids correlates most with the liquid dosing duration. From the Pearson correlation, we can observe that the liquid *1050* shows the most negative linear correlation (Pearson correlation = -0.2) with the dosing durations per liquid kg. This could be interpreted as follows: the dosing of liquid *1050* is generally faster than the normal dosing duration per kg. The most positive linear correlation (Pearson correlation = 0.1) can be observed for the liquids *1455*, *5721* and *1020*, which can be interpreted as the 'slower' liquids to dose (seconds per kg).



(A) NM1_step durations per liquids amount rounded to nearest 25 kg.



(B) Pearson correlations of liquid ingredient percentage features and NM1_step durations per kg liquid.

FIGURE 6.8: Relation between liquid amounts and the NM1_step durations.

Besides looking at the ingredient amount, the number of liquid dosages was evaluated too. The number of liquid dosages also correlate with the mixing duration, see Figure 6.9. However, the correlation is less strong than the correlation with the liquid amount.

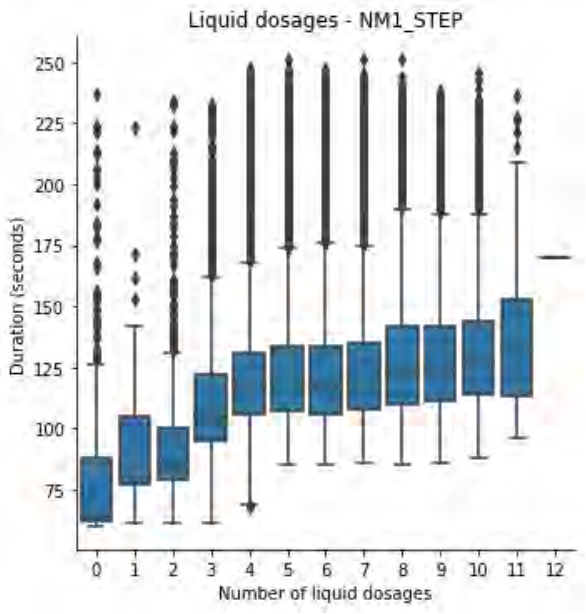
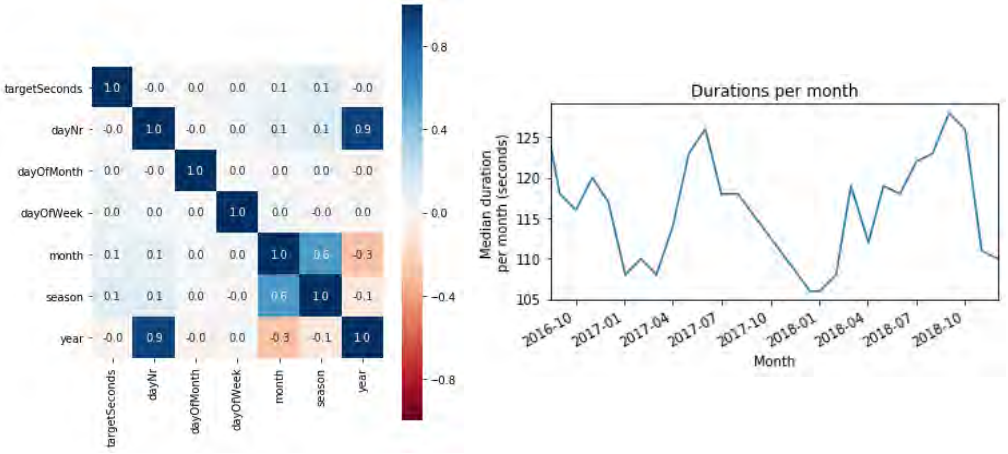


FIGURE 6.9: NM1_step durations per number of liquid dosages.

The NM1_step durations and the weather features show little correlation, see Figure A.3 of the Appendix. Only a small correlation between the NM1_step durations and the temperature (Pearson correlation = 0.1) and the humidity (Pearson correlation = -0.1) can be observed.

However, despite the small Pearson correlations between the time features and the NM1-_step durations (Figure 6.10A), a trend can be observed over time, see Figure 6.10B. A non-linear model may be appropriate to extract such a relationship.

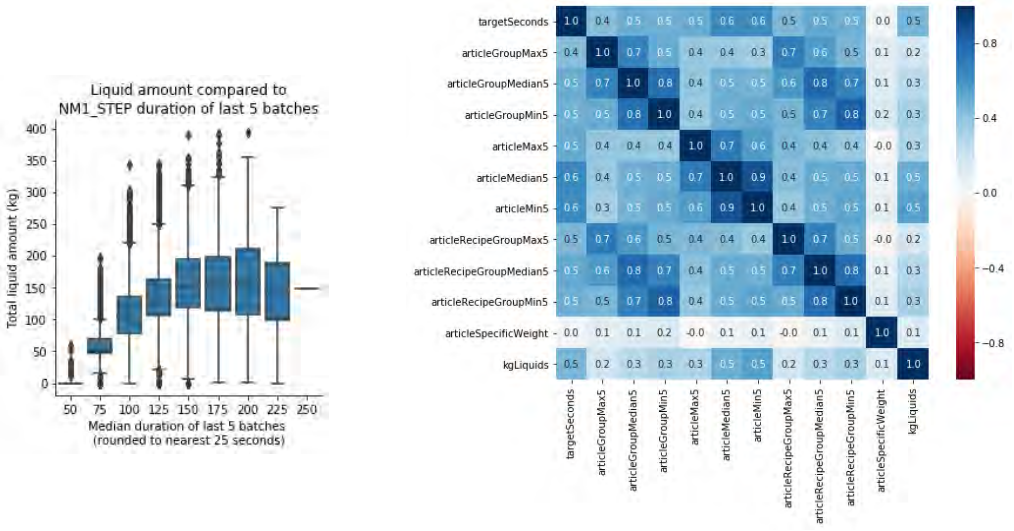


(A) Pearson correlation between time features and NM1_step durations (targetSeconds).

(B) Median NM1_step duration per month.

FIGURE 6.10: Relation between time features and the NM1_step durations.

Finally, the article features show a correlation with the NM1_step durations, see Figure 6.11B. However, the article features show a correlation with the liquid amount too. Therefore, the article features might not be as important as Figure 6.11B suggests because it resembles the relation of the liquid amount. The relation between the liquid amount and the median NM1_step duration of the last five batches is shown in Figure 6.11A.



(A) Median NM1_step duration of last 5 batches compared to the liquid amount.

(B) Pearson correlation between article features and NM1_step durations (targetSeconds).

FIGURE 6.11: Article features.

6.5 NM1_Los_step

The fifth target variable is NM1_Los_step: the duration to completely empty the mixer unit. In Section 4.3, a shift in the NM1_Los_step durations could be observed (Figure 4.16). This shift results in a positive linear correlation (Pearson correlation = 0.8) between the NM1_Los_step durations and the time features *year* and *dayNr*, see Figure 6.12.

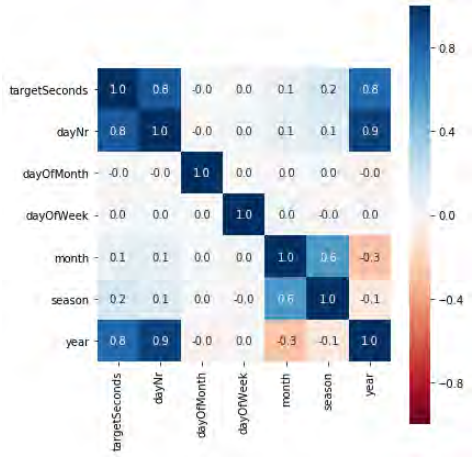


FIGURE 6.12: Pearson correlations of time features and NM1_Los_step durations.

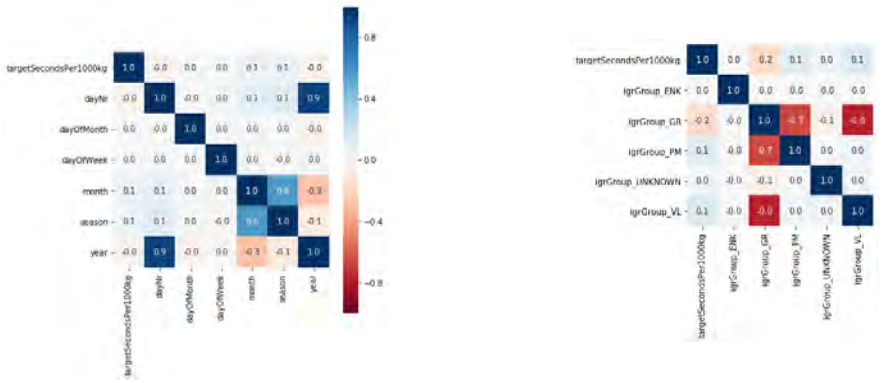
Since the spread in the NM1_Los_step durations is small over time (less than 10 seconds) and there is no correlation found between the NM1_Los_step durations and the other

features, we conclude that this production duration is not the focus of this research. The proposed model will be a simple time-based model that calculates the median duration of the last B batches.

6.6 MML_AFV_TR

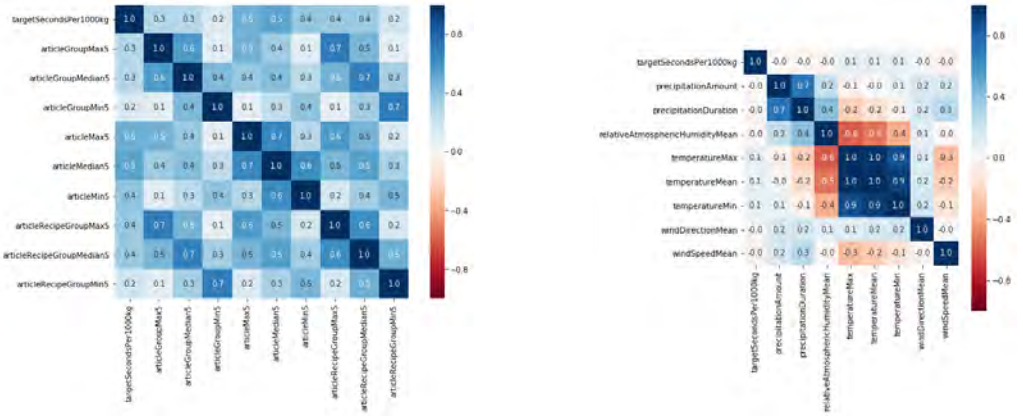
The sixth target variable is MML_AFV_TR: the duration to empty the bunker below the mixer. In Section 4.3 was already shown how the MML_AFV_TR duration correlates with the batch size, which has a Pearson correlation of 0.8. This high correlation is not surprising, because the transportation system below the bunker contains a screw conveyor, which automatically implies a correlation to the batch size.

Besides the correlation to the batch size, another correlation (Pearson correlation = -0.1) is not surprising: the dependence on the speed of the screw conveyor. However, this correlation is small because the speed of the screw conveyor is almost always (97%) set to 93%, 95% or 100%, which does not result in a much different MML_AFV_TR duration. In addition, the time features *month* and *season* show little linear correlation (Pearson correlation = 0.1) with the MML_AFV_TR durations per 1,000 kg.



(A) Time features. (B) Ingredient group features.

FIGURE 6.13: Pearson correlation between features and MML_AFV_TR durations per 1,000 kg (targetSecondsPer1000kg).



(A) Pearson correlation article features. (B) Pearson correlation weather features.

FIGURE 6.14: Relation between the MML_AFV_TR durations per 1,000 kg (targetSecondsPer1000kg) and the article and weather features.

The ingredients features show some correlation with the MML_AFV_TR durations. According to Figure 6.13B, including more raw materials (*igrGroup_GR*) results in a smaller duration per 1,000 kg (Pearson correlation = -0.2), and more pre-mixes (*igrGroup_PM*) or liquids (*igrGroup_VL*) results in a larger duration per 1,000 kg (Pearson correlation = 0.1). The Pearson correlations of individual ingredients with the MML_AFV_TR durations per 1,000 kg are shown in Figure A.4 in the Appendix.

Only the temperature features of the weather features show some linear correlation with the MML_AFV_TR durations per 1,000 kg (Pearson correlation = 0.1), see Figure 6.14B. In addition, the MML_AFV_TR durations per 1,000 kg show correlation (Pearson correlation = 0.5) with the median duration per 1,000 kg of the last 5 batches, see Figure 6.14A.

6.7 MML_AFV_TR_NADRAAI

The seventh target variable is MML_AFV_TR_NADRAAI: the duration to transport the product of the GML to the next destination. As already mentioned in 4.3, the MML_AFV_TR_NADRAAI durations depend on the location of the destination (feature *toStorageUnitNameLast*). When the MML_AFV_TR_NADRAAI durations are plotted over time for each individual destination, we can observe some shifts in the durations, see Figures 6.15 and 6.16. These shifts are most likely due to some changes in the settings of the transportation system.

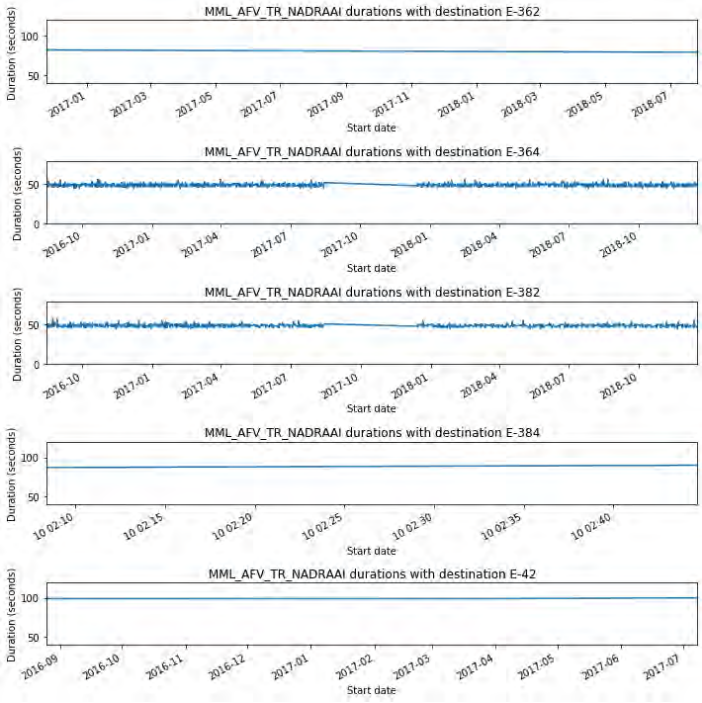


FIGURE 6.15: MML_AFV_TR_NADRAAI durations over time per storage destination.

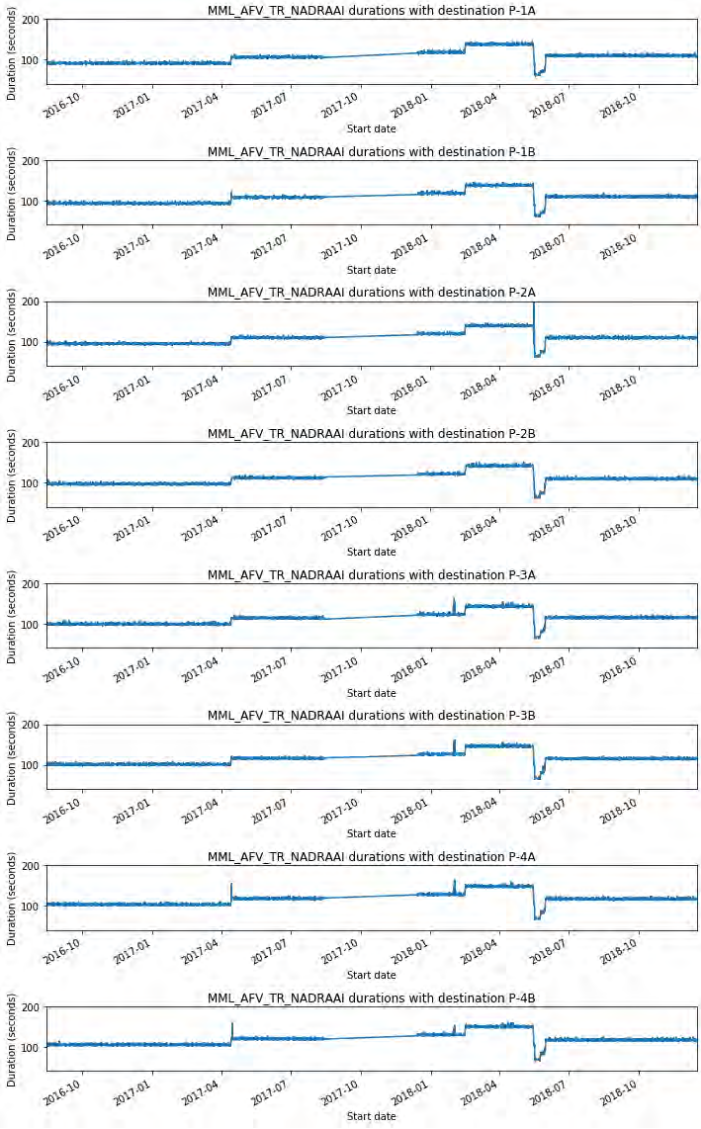


FIGURE 6.16: MML_AFV_TR_NADRAAI durations over time per press destination.

Since the spread in the MML_AFV_TR_NADRAAI durations is small over time (less than 10 seconds) and there is no correlation found between the MML_AFV_TR_NADRAAI durations and other features than the destination, we conclude that this production duration is not the focus of this research. The proposed model will be a simple time-based model that calculates the median duration of the last B batches of the same destination.

6.8 PL_PO1_slope

The eighth target variable is PL_PO1_slope: the pressing speed of 1,000 kg after finishing the warm-up period at press line 1. Since pressing is a continuous process, after the warm-up period we expect the PL_PO1 duration to be linearly dependent on the batch size, see Figure 4.22. Therefore, we expect the PL_PO1_slope values to be uncorrelated with the batch size, which is indeed the case (Pearson correlation = 0), see Figure 6.20B.

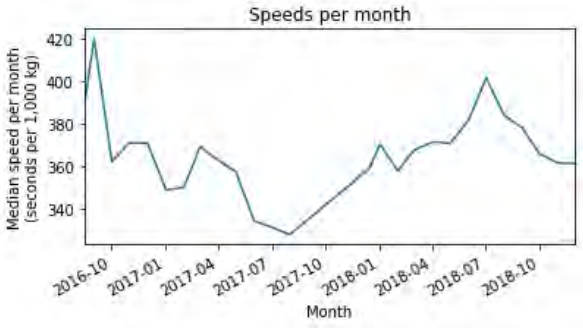
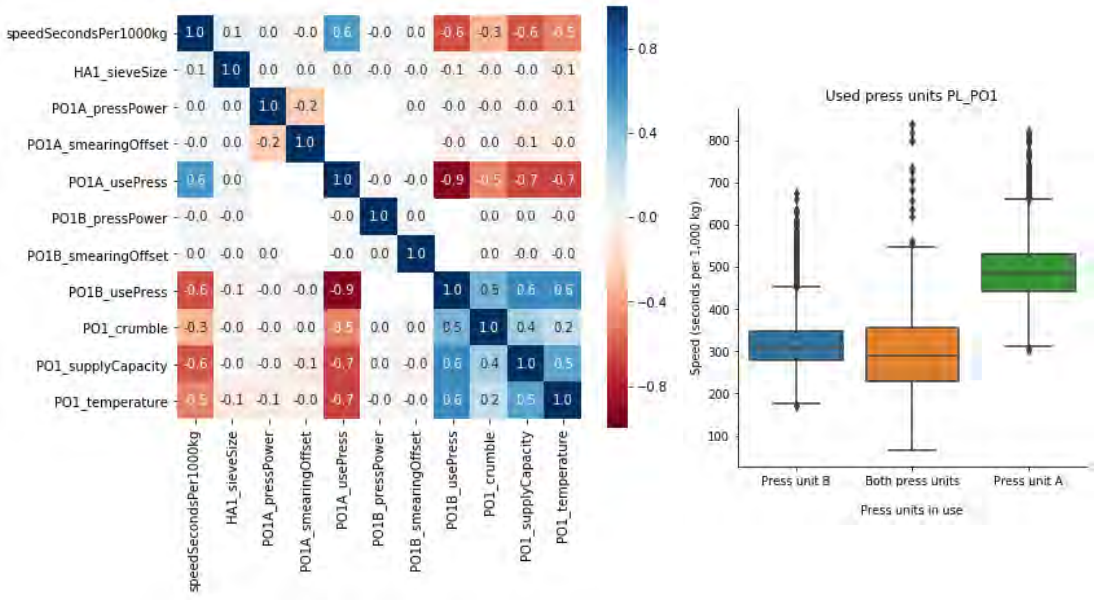


FIGURE 6.17: Median PL_PO1_slope value per month over time.

The PL_PO1_slope values show a small correlation (Pearson correlation = 0.1) with the time features *dayNr* and *year*. Figure 6.17 shows the median PL_PO1_slope values per month over time. This figure shows that the median PL_PO1_slope values per month are not constant. This could be caused by a time effect. However, it could also be caused by the decomposition of the press durations in Section 4.3.2, which could result in large fluctuations at the beginning of the dataset due to the small number of observations.

The setpoint features show correlation with the PL_PO1_slope values, see Figure 6.18A. As already was mentioned in Section 4.3, the pressing speed depends on which press units are used, see Figure 6.18B. This can also be observed from the Pearson correlation of the features *PO1A_usePress* (0.6) and *PO1B_usePress* (-0.6) in Figure 6.18A. However, approximately 87% of the batches only use press unit B.

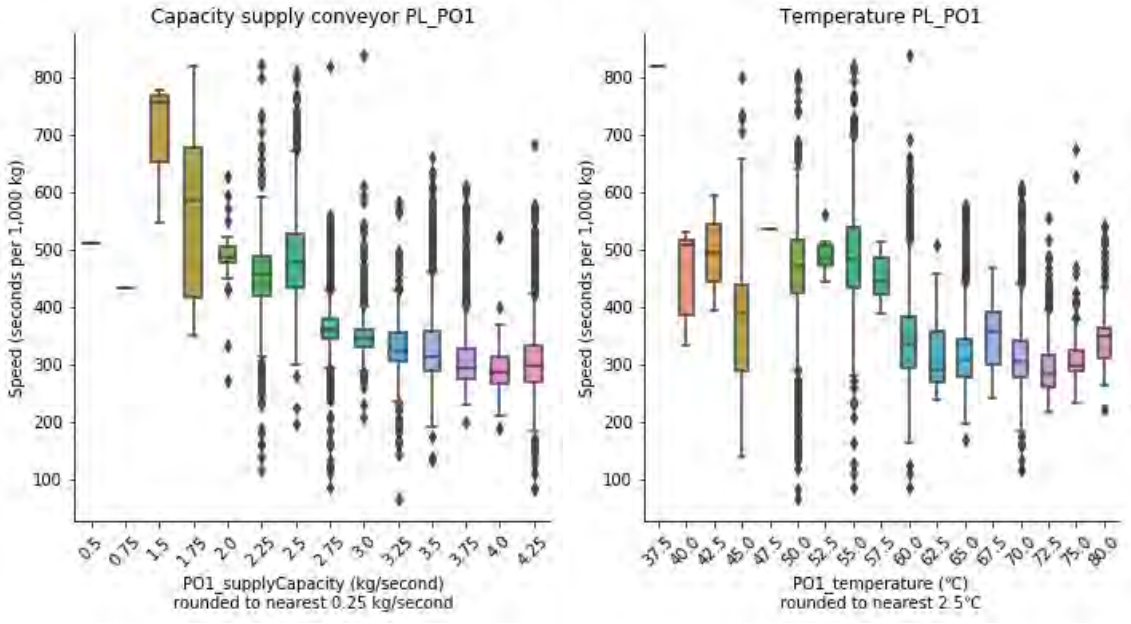


(A) Pearson correlation setpoint features.

(B) Boxplot of PL_PO1_slope values per combination of used press units.

FIGURE 6.18: Relation between the PL_PO1_slope values (speedSecondsPer1000kg) and the setpoint features.

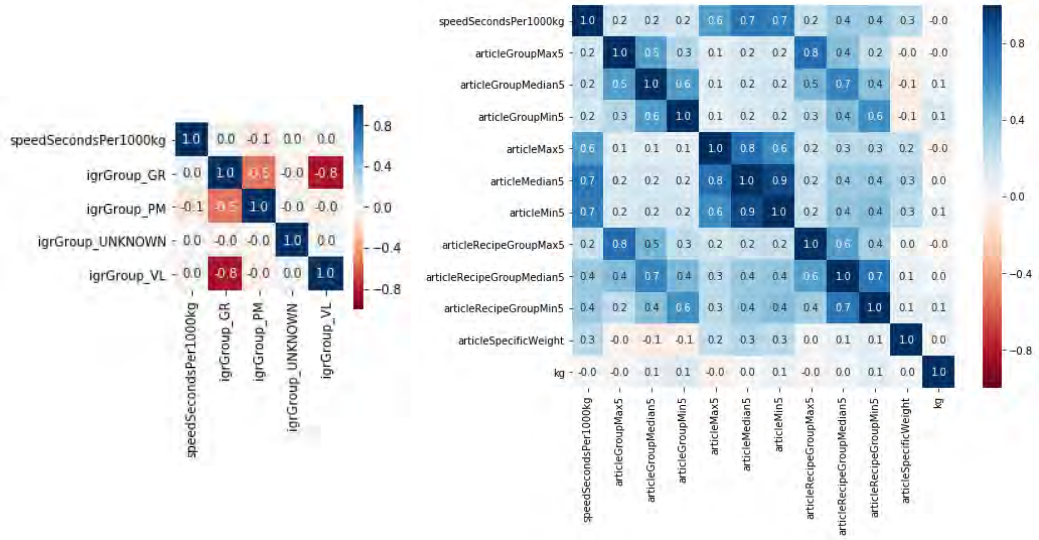
The capacity of the supply conveyor and the temperature correlate with the pressing speed too. This can also be observed from Figures 6.19A and 6.19B respectively.



(A) Supply capacity. (B) Temperature.

FIGURE 6.19: Relation between PL_PO1_slope values and setpoint features.

The correlation between the PL_PO1_slope values and the use of the crumbler and the sieve size of the hammer mills are probably caused by the correlation with which press units are used, the selected supply capacity and the selected temperature, see Figure 6.18A.



(A) Ingredient group features. (B) Article features.

FIGURE 6.20: Pearson correlation between PL_PO1_slope values and features.

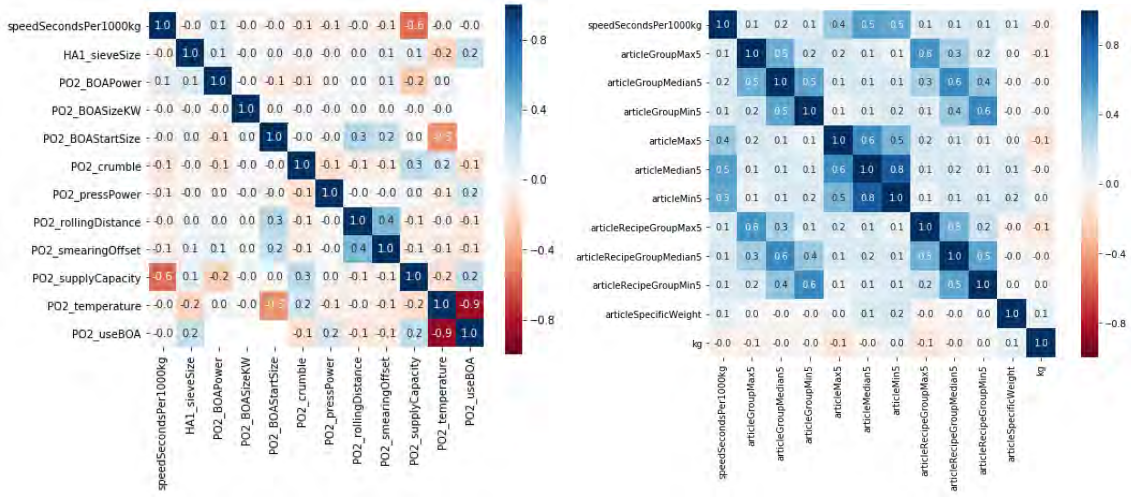
The ingredient percentage features that show small negative correlation (Pearson correlation = -0.2) with the PL_PO1_slope values are *1007*, *4455* and *979*. The ingredient percentage feature that shows the most positive correlation (Pearson correlation = 0.3) with the PL_PO1_slope values is *7441*, see Figure A.5 in the Appendix. The ingredient groups show little correlation with the PL_PO1_slope values, see Figure 6.20A. Only premixes show little negative correlation (Pearson correlation = -0.1).

No correlation (Pearson correlation = 0) is found between the weather features and the PL_PO1_slope values. Therefore, the weather features are not likely to be useful in the proposed model.

Finally, the PL_PO1_slope values are linearly correlated (Pearson correlation = 0.7) with the median PL_PO1_slope values of the last 5 batches. However, these features show also a correlation with the setpoint and ingredient features because similar kind of setpoints and ingredients are commonly used to produce the same article.

6.9 PL_PO2_slope

The ninth target variable is PL_PO2_slope: the pressing speed after finishing the warm-up period at PL 2. PL 2 has only one press unit, which is always used. Like at PL 1, PL_PO2_slope values are linearly correlated to the median speed of the previous 5 batches, too, see Figure 6.21B. In contrast to PL 1, PL_PO2_slope shows no correlation with the temperature, see Figure 6.21A.

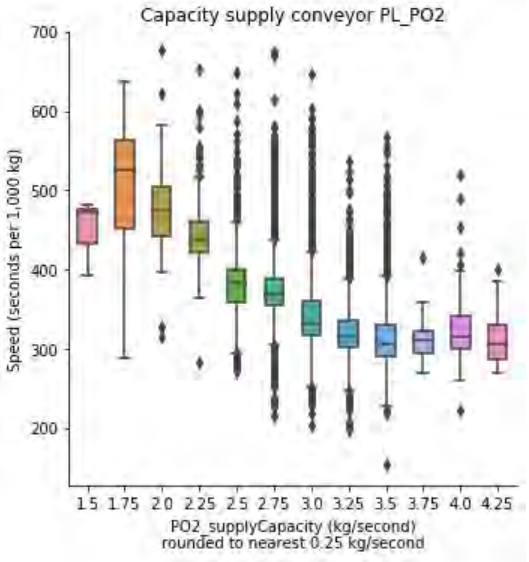


(A) Pearson correlation setpoint features.

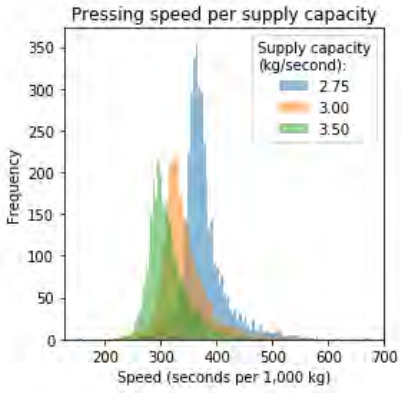
(B) Pearson correlation article features.

FIGURE 6.21: Relation between the PL_PO2_slope values (speedSecondsPer1000kg) and the setpoints and article features.

The PL_PO2_slope values mostly depend on the supply capacity (Pearson correlation = -0.6), see Figure 6.21A and Figure 6.22A. The supply capacity is mostly set to 2.75 (30%), 3.00 (24%) or 3.50 (23%) kg/second, which explains the shape of the histogram (Figure 4.27) in Section 4.3, see Figure 6.22B.



(A) PL_PO2_slope values per supply capacity (rounded to nearest 0.25 kg/second).

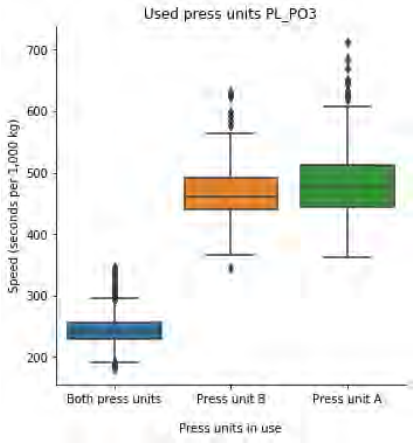


(B) PL_PO2_slope histograms for the most occurring supply capacities.

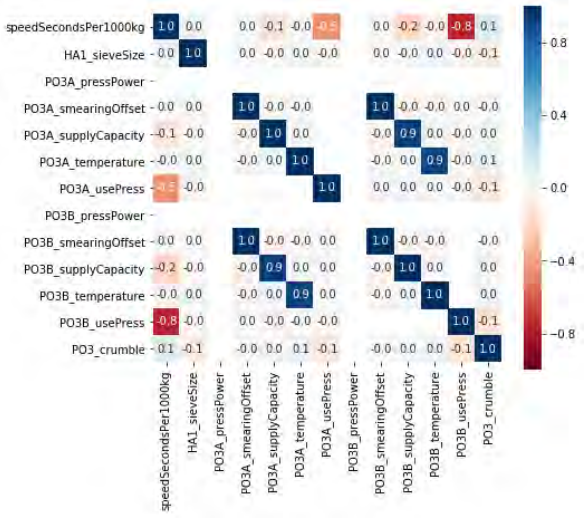
FIGURE 6.22: Relation between the PL_PO2_slope values and the supply capacity.

6.10 PL_PO3_slope

The tenth target variable is PL_PO3_slope: the pressing speed of 1,000 kg after finishing the warm-up period at press line 3. PL 3 has two parallel press units. As expected, the PL_PO3_slope values mostly depend on how many press units are used, see Figures 6.23A and 6.23B. When both press units are used, the time it takes to press is approximately twice so small as when only one press unit is used.



(A) PL_PO3_slope values per combination of used press units.



(B) Pearson correlation between PL_PO3_slope values and setpoint features.

FIGURE 6.23: PL_PO3_slope mostly correlate with which press units are used.

Furthermore, the PL_PO3_slope values are both correlated with the supply capacity to press unit A (Pearson correlation = -0.1) and the supply capacity to press unit B (Pearson correlation = -0.2). Again, the correlation to the use of the crumbler is most likely caused

by its correlation with the supply capacity. Other features show little correlation with the PL_PO3_slope values.

6.11 KO3_idle_time

The eleventh target variable is KO3_idle_time: the cooling duration at press line 3. This cooler works based on the amount of product that is inside the cooler. When there is more product than a certain level, the cooler empties. Since it is part of a continuous process, the duration depends both on the speed of the pressing process as the availability of units after the cooler.

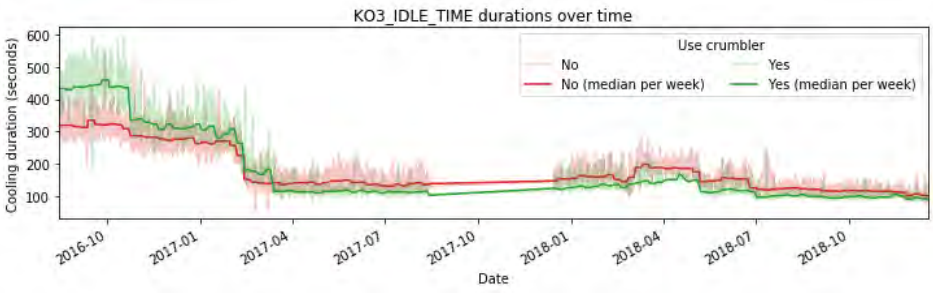


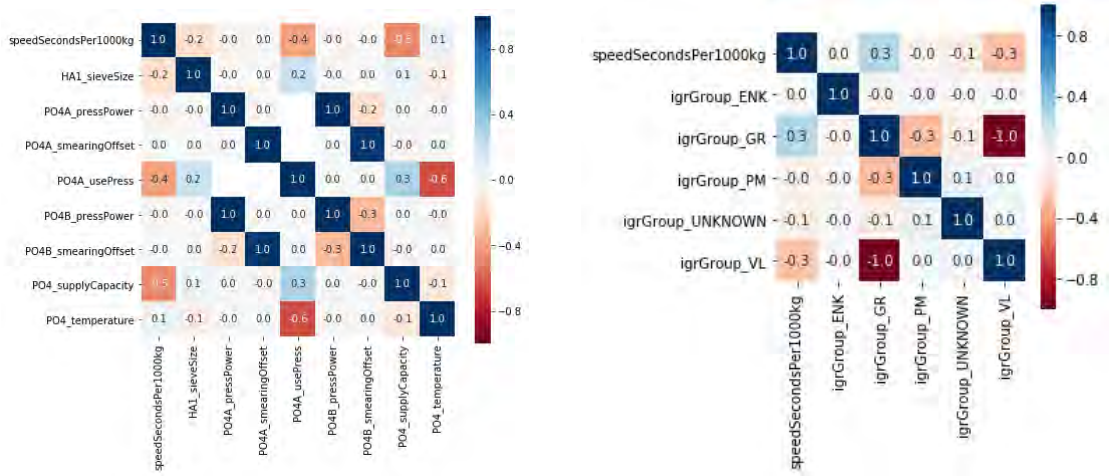
FIGURE 6.24: KO3_idle_time durations over time including whether the crumbler will be used or not.

Since the supply capacity correlates with the use of the crumbler and the use of the crumbler indicates which unit will be used after the cooler, the cooling durations are likely to depend on the use of the crumbler. Therefore, the KO3_idle_time durations are plotted over time for both using and not using the crumbler, see Figure 6.24. The figure confirms the suspicion of a correlation with the use of the crumbler. The Pearson correlation between the KO3_idle_time durations and the use of the crumbler is 0.2.

Since March 2017, there is not much spread in the KO3_idle_time durations. In addition, no correlation with other features was found. Therefore, we conclude that this production duration is not the focus of this research. The proposed model will be a simple time-based model that calculates the median duration of the last B batches per usage of the crumbler.

6.12 PL_PO4_slope

The last target variable is PL_PO4_slope: the pressing speed of 1,000 kg after finishing the warm-up period at press line 4. PL 4 has two press units in series, like PL 1. According to Figure 6.25A, the PL_PO4_slope values mostly correlate which press units are used and the selected supply capacity. These relations are visualized in Figures 6.26A and 6.26B.

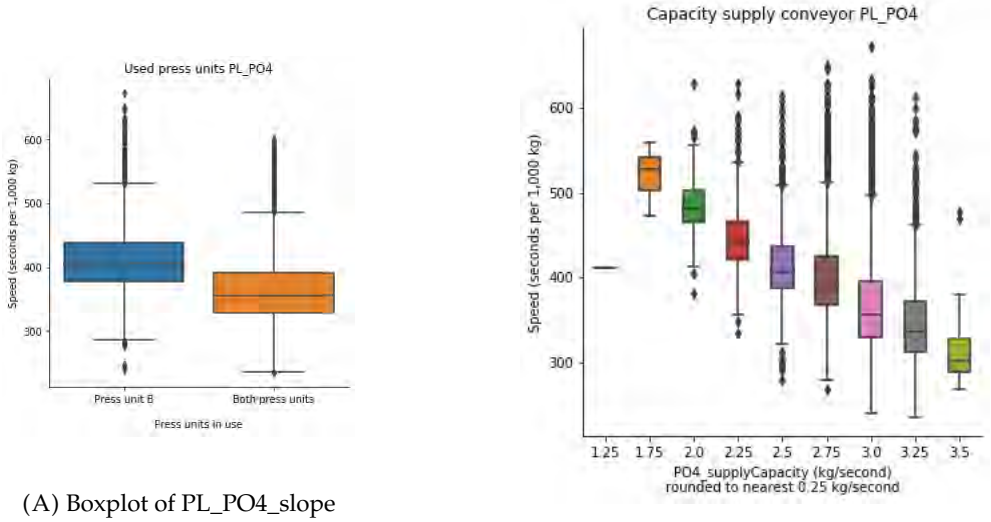


(A) Setpoint features and PL_PO4_slope values (speedSecondsPer1000kg).

(B) Ingredient group features and PL_PO4_slope values (speedSecondsPer1000kg).

FIGURE 6.25: Pearson correlations.

According to Figure 6.25B, the ingredient groups raw materials and liquids correlate with the PL_PO4_slope values with 0.3 and -0.3 respectively. For more details on specific ingredients, see Figure A.6 in the Appendix.



(A) Boxplot of PL_PO4_slope values per combination of used press units.

(B) PL_PO4_slope values compared to the supply capacity.

FIGURE 6.26: Relation between the PL_PO4_slope values and the setpoints.

7 Methods

The machine learning models and evaluation methods that will be used in this research are discussed in this section. Since there is not one machine learning model that works best in every situation, as addressed by the No Free Lunch Theorem [25], several machine learning models are tested and the best model for our purpose is selected.

In this research, 12 *target variables*, or *response variables*, are defined, see Table 2.1. Note that the target variables for the pressing durations at the press lines are changed to the pressing speeds during period B, as mentioned in Section 4.3.2, see Table 4.3. Since it could be the case that a model estimates one target duration accurately, but another target not accurate at all, separate models are created for the prediction of the target variables. The features described in the previous sections are used for this purpose, which are also called *explanatory variables*, or *attributes*.

In general, prediction problems can be split into two classes: classification problems and regression problems. The target variable in a classification problem consists of two or more classes. For example, if you would like to predict if the produced article is mash, crumble or pellets, you could define this as a classification problem in which the target variable consists of the classes mash, crumble and pellets. In contrast, the target variable in regression problems is a numerical value. This is the case in this research because all production durations are numerical (positive) values. Therefore, this research aims to give a solution to 12 regression problems.

Since the estimation of production durations is a regression problem, the focus of this section will be on regression models, also called *regressors*. The regressors will be compared with each other on a test sample set, further denoted as *test samples* or *test set*, which are not used for the creation of the models. In contrast to the *training samples* or *training set*, which are the observations on which the regressors' estimations are based. The differences between the estimated values and the observed values are commonly called *residuals*.

7.1 Linear Regression

The first regressor that is evaluated is Linear Regression. The idea is to fit a linear function $\hat{y}$ to the observations y using explanatory variables x. The model "learns" the weight w_j of each feature j, and the value of the intercept b. The prediction $\hat{y}_i$ of production duration y_i is given by the sum of the weighted features, $w_j x_{ij}$, and intercept b, see Equation (7.1).

$$\hat{y}_i = b + \sum_{j=1}^{m} w_j x_{ij} \tag{7.1}$$

The weights w are "learned" by optimizing an objective function on n training samples, which is in case of ordinary Linear Regression the sum of squared residuals r, which is the sum of the squared difference between the estimated values $\hat{y}$ and the observed values y,

see Equation (7.2).

$$\begin{aligned} &\text{Objective} = min \sum_{i=1}^{n} {r_i}^2 \\ &r_i = y_i - \hat{y}_i \end{aligned} \tag{7.2}$$

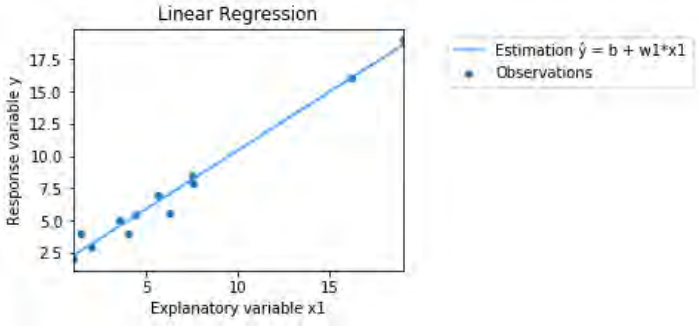


FIGURE 7.1: Example estimation of y by $\hat{y}$ using Linear Regression.

An example of a Linear Regression function with one explanatory variable x_1 is given in Figure 7.1. There is the possibility of dropping the intercept (setting b to zero) by changing the hyperparameter *fit_intercept* from True to False, see Table 7.1. Both possibilities will be tested in this research. Note that a hyperparameter is a predefined setting of the model. This is different from a weight parameter w that is trained by optimizing the objective function.

Hyperparameter	**Description**	**Default value**	**Evaluated values**
fit_intercept	Whether an intercept is fitted or not.	True	True, False

TABLE 7.1: Hyperparameter settings of Linear Regression.

7.2 Robust Linear Regression

The second model is much related to the first one. However, a Linear Regression model is known to be sensitive for outliers, see the left plot in Figure 7.2 for an example. To make the model more *robust* to outliers, the least squares, which are minimized in Linear Regression, are iteratively reweighted using a robust criterion estimator [26]. Which robust criterion is used is optional in the model, see Table 7.2 for the hyperparameter M.

The default value of M is the Huber's loss function [26], which results in squared values for small residuals ($\leq t$) and absolute values for large residuals ($> t$) [27], see Equation (7.3). The default value of t is 1.345, which is also used in this research.

$$\begin{aligned} &\text{Objective} = min\ L \\ &L = \begin{cases} \sum_{i=1}^{n} {r_i}^2, & \text{if } r_i \leq t \\ \sum_{i=1}^{n} |r_i|, & \text{if } r_i > t \end{cases} \\ &r_i = y_i - \hat{y}_i \end{aligned} \tag{7.3}$$

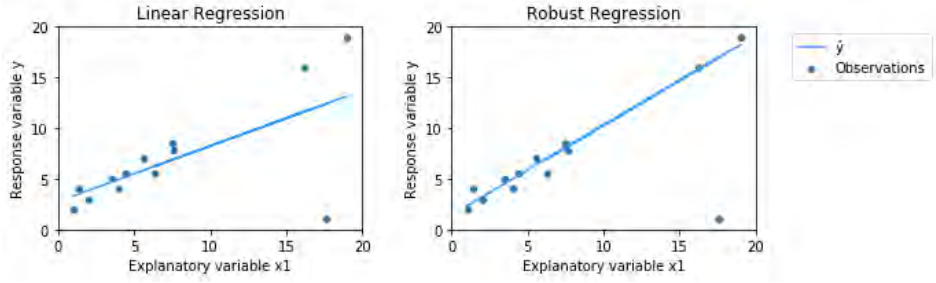


FIGURE 7.2: Example estimation of y by $\hat{y}$ using Linear Regression (left) and Robust Linear Regression (right).

The result is a more robust model because big errors have a smaller weight (absolute value) in Robust Linear Regression than in Linear Regression (squared value). Figure 7.2 shows the estimations of Linear Regression (left) and Robust Linear Regression (right) in an example with an outlier (observation (17.6; 1)).

Hyperparameter	Description	Default value	Evaluated values
M	The robust criterion function for downweighting outliers [26].	HuberT	HuberT

TABLE 7.2: Hyperparameter settings of Robust Linear Regression.

This model is used twice in this research. First of all, it is used for the separation of periods A and B of the pressing durations at the press lines, as was described in Section 4.3.2. In this case, only the batch size is used as an explanatory variable (x_1). The estimated intercept (b) is taken as period A; the warm-up period. The new speed targets of the press lines are created by subtracting the warm-up period (b) from the total duration (y) divided by the batch size (x_1), as was explained in Section 4.3.2, see Table 4.3.

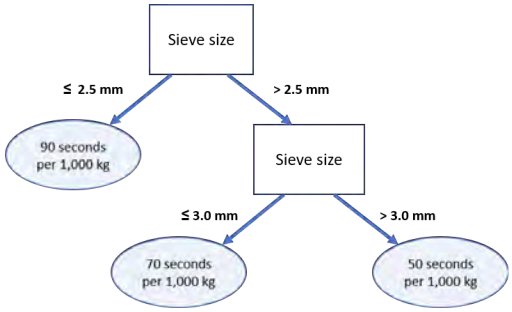
In addition, this model is evaluated for the estimations of all target variables; the speed of the press lines (Table 4.3) and the total duration of the other production durations of Table 2.1. Hence, it is possible that this model is used twice in the estimation of the pressing durations at the press lines.

7.3 Decision Trees

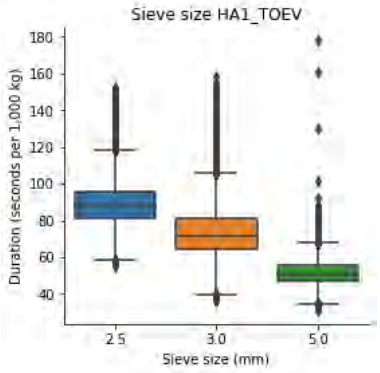
The previous models are linear models. However, non-linear relations might be present in the data for which another kind of models could perform more accurately. Therefore, some state-of-the-art non-linear models are evaluated too. Many of these models are based on the concept of Decision Trees. Therefore, we will first describe what Decision Trees are and how they can be used for regression problems.

Decision Trees are created by a set of if-then-else decision rules. An example is given for the estimation of the HA1_TOEV durations per 1,000 kg in Figure 7.3A. These simple decision rules are able to separate the three sieve sizes of the hammer mill (Figure 7.3B). Hence, Decision Trees can be easily interpreted, which is one of the biggest advantages of this model.

Another situation in which a Decision Tree could perform well is when a shift in the production durations occurs in time, for example, due to the cleaning of the machine or a change in the non-batch related machine settings. The Decision Tree would be able to



(A) Example estimation by a Decision Tree, with the hammer mill sieve size as feature.



(B) Boxplot for each used sieve size.

FIGURE 7.3: HA1_TOEV durations per 1,000 kg.

separate the data points before and after this point in time easily. In contrast to a Linear Regression model, which needs this information to be explicit, for example by a binary feature (1=before point in time, 0=after point in time).

The usage of a set of decision rules results in a step-wise function of an explanatory variable, see the blue and green line in Figure 7.4. How many steps are created depends on the depth of the Decision Tree, i.e., the number of decision rules. When the maximum depth of a Decision Tree is not limited, a decision rule could be included for the separation of each individual data point. This is the biggest disadvantage of Decision Trees: it is prone for *over-fitting*, which means that it leads to "discovery" of effects that are actually spurious [28], [29], see the green line in Figure 7.4. In that case, the model is not able to generalize enough, which results in good estimations on the training samples, but bad estimations on test samples. However, *under-fitting*, not discovering actual effects, should be avoided too. For example, the step-wise blue line in Figure 7.4 could fit the data points better if the step size becomes smaller, i.e. the number of steps increases. Therefore, hyperparameter tuning is very important to find the hyperparameters that fit the data points best.

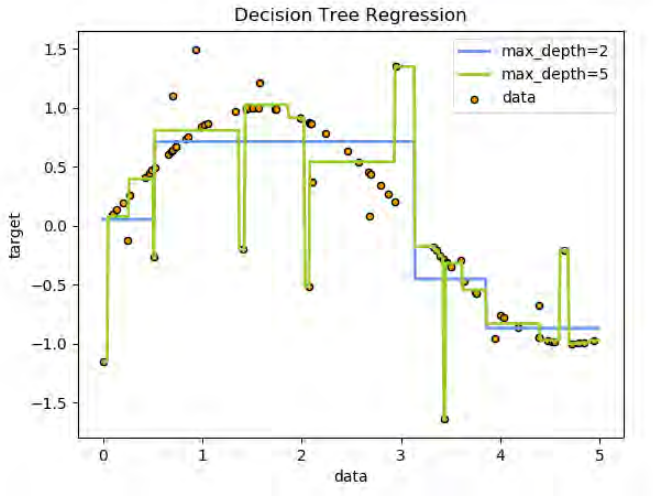


FIGURE 7.4: Example estimation of target y using Decision Tree Regression [30].

Other hyperparameters of Decision Trees consist of the number of samples in each endpoint in the tree, a *leaf*, and the number of samples needed for an additional decision rule,

a *split*. Tuning the hyperparameters is important to make the model accurately estimate future production durations.

7.3.1 CART

How the decision rules, or *splits*, are created depends on the selected Decision Tree algorithm. One commonly used algorithm is *Classification and Regression Trees* (CART) [31], which, as it states, can be used for both classification and regression problems. This algorithm is used by many tree-based models and will, therefore, be described in this section.

The CART algorithm, introduced by Breiman et al. in 1984, is characterized by binary splits. Each internal *node* has exactly two outgoing *edges* [31], [32], as in the example of Figure 7.3A is shown. Each split is made on a single explanatory variable, for example, sieve size. However, there is no limitation on the number of explanatory variables used in the complete tree. Hence, it is possible to split, for example, first on the sieve size, then on the year, and finally on the sieve size again.

The predictions of the CART Decision Tree are obtained by calculating the weighted mean over the samples in each *leaf*. Since the Decision Tree of the example in Figure 7.3A consists of three leaves, the predictions will consist of three unique values: the mean over the batches using a sieve size of 2.5 mm, the mean over the batches using a sieve size of 3.0 mm and the mean over the batches using a sieve size of 5.0 mm.

The residuals of a node can be calculated by the predictions and observations in that node. The algorithm decides how to make the next split by minimizing the squared residuals of the two created leaf nodes [32]. The algorithm works in a *greedy* manner, which means that it iteratively adds best splits. Note that this approach does not guarantee an optimal tree, as sometimes worse splits should be made to be able to create a better split in the next iteration. The splitting process stops when the number of samples in the node is less than a certain minimum (*min_samples_split*), the number of samples contained in the created leaves after the split is less than a certain minimum (*min_samples_leaf*) or the maximum depth (*max_depth*) of the tree is reached. In that case, the final node is taken as a leaf node.

7.3.2 Limitations

As already mentioned, Decision Trees are prone for over-fitting. Besides that, there are other limitations to Decision Trees. First of all, a Decision Tree has a high variance. This means that the predicted values depend a lot on the training set. When the training set changes, the predictions could change a lot too.

In addition, Decision Trees have the problem of the lack of smoothness. In the example of Figure 7.3A only three different values are predicted. There is nothing in between. Hence, the predicted values are not *smooth*.

These limitations of Decision Trees are the reason that single trees are not frequently used. However, *ensembles* of Decision Trees have success in many applications [21].

7.4 Extra Trees

The first tree-based ensemble model that is used in this research, is Extra Trees [33]. This model tries to improve the accuracy of a Decision Tree and to control over-fitting [34]. This is done by the creation of multiple CART Decision Trees. Each tree is build using the whole dataset. However, the features used for a split can be limited to a random choice of a certain number of features [33], which is specified by hyperparameter *max_features*. In addition, instead of the greedy splitting approach, each split of a certain feature is made at random

[33]. This results in *extremely randomized trees*, which are able to reduce the model variance. The model combines the information to one prediction by taking the arithmetic mean over the individual Decision Trees. An advantage of this model is that the confidence interval of a prediction could be defined based on the individual Decision Trees.

How many Decision Trees are created in the model is determined by the hyperparameter *n_estimators*, see Table 7.3. In general, the larger the number of Decision Trees, the better the results, but the longer it will take [34]. Therefore, it was decided to use the default number of 100 Decision Trees, which is "large enough to ensure convergence of the ensemble effect" with all datasets used by Geurts et al. (2006) [33].

In addition, hyperparameters are included for the individual Decision Trees: the maximum depth (*max_depth*), the minimum number of samples to split (*min_samples_split*) and the minimum number of samples in each leaf (*min_samples_leaf*). Since the best hyperparameters are not known, several are tested and evaluated on a validation set to decide which hyperparameters are the best, see the overview in Table 7.3.

Hyperparameter	**Description**	**Default value**	**Evaluated values**
n_estimators	The number of created Decision Trees.	100	100
max_depth	Maximum depth of each Decision Tree.	∞	∞, 10, 20, 30, 40, 50
min_samples_split	Minimum number of data samples for split in Decision Tree.	2	2
min_samples_leaf	Minimum number of data samples in each leaf of each Decision Tree.	1	1, 10, 50, 100, 200, 500
max_features	The number of features to consider when looking for the best split in a Decision Tree.	Total number of features	Total number of features

TABLE 7.3: Hyperparameter settings of Extra Trees.

The evaluated values of *max_depth* are set to infinity, 10, 20, 30, 40 and 50 and the evaluated values of *min_samples_leaf* to 1, 10, 50, 100, 200 and 500, which are chosen to see the influence of limiting the complexity of the individual Decision Trees.

Finally, the *max_features* hyperparameter is chosen to be the total number of features, which is the default value of *max_features* in regression problems and is recommended by Geurts et al. [33]. For more details on Extra Trees, please refer to the original paper proposed by Geurts et al. in 2006 [33].

7.5 Gradient Boosting Machine

There exist other ensemble models. One variant is the Gradient Boosting Machine (GBM) [35], which uses a *boosting* technique. "The idea of boosting is to use the weak learner to form a highly accurate prediction rule by calling the weak learner repeatedly on different distributions over the training samples" [36]. In which a weak learner is defined as a model that performs "just slightly better than guessing" [37]. The reasoning is that each of these weak

learners represents a "rule-of-thumb", which are by themselves "very rough and inaccurate", but combining many "rules-of-thumb" could lead to a very accurate prediction [37].

$$\text{Objective} = min\ L + \Omega \tag{7.4}$$

First of all an objective is specified, see Equation (7.4). This objective consists of a loss function L and a regularisation term Ω. The loss function represents the error made on the observations. For the standard GBM model, the sum of squared residuals is taken, see Equation (7.5).

$$\begin{aligned} L &= \sum_{i=1}^{n} {r_i}^2 \\ r_i &= y_i - \hat{y}_i \end{aligned} \tag{7.5}$$

The seconds part of Equation (7.4) is the regularization term. The purpose of the regularisation term is to penalize complicated models, which "encourages simple models" [38]. "Simpler models tend to have smaller variance in future predictions, making prediction stable" [38].

As mentioned, the prediction of a boosting model is based on multiple simple models. In the case of standard GBM, also called Gradient Boosting Decision Trees (GBDT), CART Decision Trees are used [38], which were introduced in Section 7.3.1. We define the prediction of a single Decision Tree as f_t, then the prediction of the GBM is the sum of all (T) individual trees, see Equation (7.6) [35], [38]. This function is used to estimate all (n) observations in the training set.

$$\hat{y}_i = \sum_{t=1}^{T} f_t(x_i) \tag{7.6}$$

Hence, the objective function to be minimized contains many Decision Trees instead of numerical vectors and therefore standard optimization techniques can not be used [38]. Since it becomes "intractable" to train all Decision Trees at the same time, an iterative procedure is used [38], of which an abstract visualization is shown in Figure 7.5.

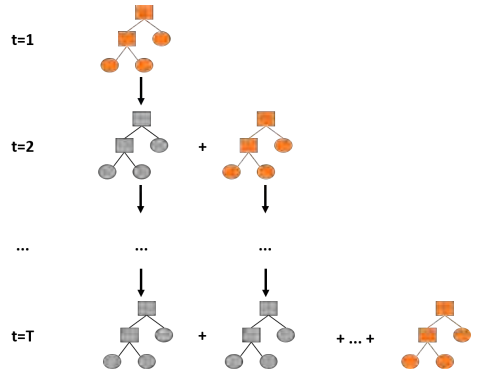


FIGURE 7.5: Abstract visualization of Gradient Boosting Machine [35], [38]. Orange: trained in current iteration, grey: reuse previous iteration.

In each iteration, a new Decision Tree is added, which aims to reduce the remaining error of the previous trees. This results in the summation of the prediction of the previous trees

(1, ..., $t-1$) plus the prediction of the current tree (t), see Equation (7.7) [38]. The objective function of created tree in iteration t is shown in Equation (7.8). Decision Trees are added until T trees are created, see Figure 7.5.

$$\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i) \tag{7.7}$$

So far, we did not specify the regularisation term Ω. As mentioned, this term indicates how complicated the created Decision Trees are. Since only one Decision Tree is build per iteration, the regularisation term of the previous build Decision Trees are constant in the objective function of the current iteration, see Equation (7.8).

$$\begin{aligned} \text{Objective}^{(t)} =& min \sum_{i=1}^{n} (y_i - \hat{y}_i^{(t)})^2 + \sum_{i=1}^{t} \Omega(f_i) \\ =& min \sum_{i=1}^{n} (y_i - (\hat{y}_i^{(t-1)} + f_t(x_i)))^2 + \sum_{i=1}^{t} \Omega(f_i) \\ =& min \sum_{i=1}^{n} ((y_i - \hat{y}_i^{(t-1)}) - f_t(x_i))^2 + \Omega(f_t) + constant \end{aligned} \tag{7.8}$$

There are multiple regularisation methods. Commonly used regularisation methods are described by the following function, see Equation (7.9). The function contains three elements. The first part (γB) penalizes trees with many leaves; with B leaves and weight γ. The second part is the L_1-regularization on the values of the leaves, in which α is the weight of regularization and w_j is the value of leaf j. Finally, the last part is the L_2-regularization term, which is very similar to the L_1-regularization, but now it takes the square instead of the absolute value of the leaf value and its weight parameter is λ. If and which regularization is used can be chosen by setting the hyperparameters γ, α and λ.

$$\Omega(f_t) = \gamma B + \alpha \sum_{j=1}^{B} |w_j| + \frac{1}{2}\lambda \sum_{j=1}^{B} {w_j}^2 \tag{7.9}$$

For standard GBM, as described by Friedman et al. in 2001 [35], no regularization was implemented, which is the same as setting these three hyperparameters to zero. Setting to zero simplifies the objective function (Equation (7.4)) to only the minimization of the loss function L.

The objective function is defined above. Note that the predictions of many Decision Trees are summed together in the GBM. For the optimization of the GBM, the Decision Tree that is created in an iteration should make the best contribution to the whole GBM model. Steepest descent is used for that purpose [35]. Steepest descent calculates the gradient of the objective function. Since the objective is a minimization function, the negative gradient is taken as the direction to which the parameters of the *weak learner* are updated [35]. The splits of the new Decision Tree should be created in such a way that it results in leaf values most parallel to the negative gradient. Therefore, the defined response variable, containing "pseudo responses", of the Decision Tree created in iteration t, can be given by Equation

(7.10) [35].

$$\begin{aligned} &\text{Pseudo responses}^{(t)} : \{-g_t(x_i)\}_{i=1}^{N} \\ &\text{with } N \text{ training samples} \\ &\text{and } g_t \text{ the gradient of the objective function in iteration } t-1 \end{aligned} \tag{7.10}$$

7.6 XGBoost

XGBoost [21], "eXtreme Gradient Boosting", is a boosting model inspired by the Gradient Boosting Machine from Friedman et al. The initial version of the model was released in 2014 as a Python and R module [39]. The corresponding paper was published in 2016 [21]. According to the paper, it has been successful in many applications and many winners of machine learning competitions used XGBoost [21].

As mentioned, regularisation was not implemented in standard GBM. This was introduced by XGBoost, and is therefore sometimes called "regularized boosting" [21]. The default implementation of XGBoost has $\gamma = 0$, $\alpha = 0$ and $\lambda = 1$ (Equation (7.9)) [40], which only includes the L_2-regularisation term.

Another difference that was made by XGBoost compared to the original implementation of Gradient Boosting Machine is that XGBoost uses parallel computation. Since the Decision Trees are built in series, they cannot be built simultaneously. However, how the Decision Tree grows could be parallelized. According to Chen et al., "the system runs more than ten times faster than existing popular solutions on a single machine and scales to billions of examples in distributed or memory-limited settings" [21].

7.7 Light Gradient Boosting Machines

Recently, another Gradient Boosting framework was introduced by Ke et al. in 2017: LightGBM [41]. It was developed to improve the efficiency and scalability of the Gradient Boosting model, and they succeeded. According to Ke et al., "LightGBM can accelerate the training process by up to over 20 times while achieving almost the same accuracy". Two novel techniques were used, which will be described in this paragraph.

Firstly, Ke et al. noticed that training samples with larger gradients will contribute more to the creation of the Decision Trees. To speed up the learning process, we can downsample data instances. Ke et al. proved that random downsampling data instances with small gradients (below the top percentiles or under a certain threshold) can lead to a more accurate gain estimation than uniformly random sampling [41]. This technique is called *Gradient-based One-Side Sampling* (GOSS) [41]. However, this technique is not used in the default implementation of LightGBM, and it is therefore not used in this research.

The second technique, *Exclusive Feature Bundling* (EFB) [41], is part of the default implementation and will, therefore, be used in this research. This technique bundles exclusive features to a single feature, which is specifically useful in a sparse feature space [41]. For example, the ingredient features in the estimation of production durations are often zero, because many ingredients are contained in only a part of the produced articles. According to Ke et al. combining exclusive features reduces the complexity without hurting the accuracy [41]. However, the *graph colouring problem* [42] could be reduced to this feature bundling problem. The *graph colouring problem* is known to be NP-hard, which means that it is impossible to find an optimal solution in polynomial time in general. But, there are *greedy* algorithms available for the *graph colouring problem*, which can be used to efficiently give a

good approximation in the feature bundling problem. For more information on EFB and LightGBM, please refer to the original paper of Ke et al. [41].

In general, Gradient Boosting models have many hyperparameters to tune. First of all, Decision Trees themselves have hyperparameters like the maximum depth of the tree (*max_depth*) and the minimum number of data samples in the leaves of the tree (*min_data_in_leaf*). In addition, the maximum number of leaves (*num_leaves*) could be specified in the LightGBM model.

Since the model is trained using gradient descent, a *learning rate* is included, which is used to shrink the updating step sizes. This learning rate is used to prevent the model from over-fitting [41]. Since the direction of the gradient specifies the created Decision Tree, shrinking the step size increases the need for more Decision Trees (*n_estimators*) because more steps are needed to reach the same distance. However, including more Decision Trees increases the training time of the model.

Finally, the regularisation term of the objective function could be specified using γ, α and λ of Equation (7.9). Which are zero by default in LightGBM, but λ is 1 by default in XGBoost.

Since the training time of LightGBM is much smaller than that of XGBoost, and they reach approximately the same accuracy [41], only LightGBM is used in this research. The following hyperparameters will be evaluated, see Table 7.4.

Hyperparameter	**Description**	**Default value**	**Evaluated values**
max_depth	Maximum depth of each Decision Tree.	∞	∞, 6, 8, 10
min_data_in_leaf	Minimum number of data samples in each leaf of each Decision Tree.	1	10, 50, 100
num_leaves	Step size shrinkage used in update to prevents over-fitting.	31	50, 100, 150, 200
n_estimators	The number of created Decision Trees.	100	100, 300, 500
learning_rate	Step size shrinkage used in update to prevents over-fitting.	0.1	0.05, 0.1, 0.2
lambda_l2	L_2 regularisation factor	0	0, 1

TABLE 7.4: Hyperparameter settings of Light Gradient Boosting Machines.

According to J. Aarshay [43], common starting point of the *max_depth* hyperparameter is 8 and general good values for the *learning_rate* are between 0.05 and 0.2. The number of leaves (*num_leaves*) should be smaller than the maximum number of leaves that are possible based on the *max_depth* hyperparameter (2^{max_depth}) to take effect. Hence, we will only evaluate $num\_leaves = 50$ for $max\_depth = 6$, because $2^6 = 64$.

To see how many Decision Trees are needed, the values 100, 300 and 500 will be tested. Finally, three values of the minimum number of data samples in each leaf are evaluated: 10, 50 and 100 samples. Since not limiting the created trees (minimum number of data samples = 1) likely results in over-fitting, some values larger than one are used. In addition, not limiting the Decision Trees could result in very big trees, which increases the running time significantly. To evaluate the effect of limiting the created trees in the number of data samples in each leaf, the values 10, 50 and 100 are used.

7.8 Artificial Neural Network

This section started with the introduction of Linear Regression, which "learns" the weights of a linear function in order to optimize its estimation of the target variable, see Equation (7.1). This function shows resemblance with an *artificial neuron*, which combines the feature values x of m features, also named *input signals*, with weights w and an intercept b, also named *bias*, using an *activation function* ψ, see Equation (7.11) and Figure 7.6.

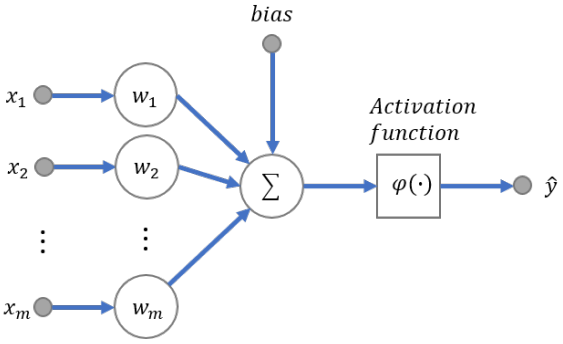


FIGURE 7.6: Abstract visualization of an artificial neuron [44].

$$\hat{y}_i = \psi(b + \sum_{j=1}^{m} w_j x_{ij}) \tag{7.11}$$

The first model containing an artificial neuron was the Perceptron, which was introduced by F. Rosenblatt in 1957 [45]. The Perceptron is used for binary classification problems (predicting two classes, for example, yes or no) in which ψ is a *threshold function*, see Table 7.5 [45]. It fits a linear function to the training samples and predicts everything above the function (≥ 0) as positive (or yes) and everything below (< 0) as negative (or no).

Later, multiple artificial neurons were used in a Multi-Layer Perceptron (MLP). This model is a class of the feed-forward Artificial Neural Networks. The MLP consists of at least three layers: an input layer, one or more hidden layers, and an output layer. A visualisation of an example of a MLP with one hidden layer with four neurons is given in Figure 7.7.

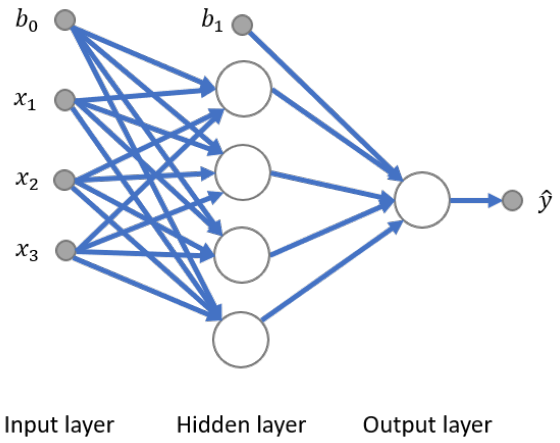


FIGURE 7.7: Example of an Multi-Layer Perceptron with 3 features (x_1, x_2 and x_3), one hidden layer with 4 neurons, a single output variable $\hat{y}$, and biases b_0 and b_1.

The input layer consists of the input signals (the features), which are combined with the bias on the input layer (b_0) in the neurons of the first hidden layer. Each neuron of the hidden and output layers works like described above in Equation (7.11). When there are multiple hidden layers present, the output of the neurons of the previous layer (plus bias) are the inputs of the next layer. Finally, the output layer combines the output of the neurons of the last hidden layer. See Figure 7.7 for a visualization. Note that the signals in the MLP go only forward to the next layer (*feed-forward*), and are connected to every neuron in a layer (*fully connected*).

Since our problem is a regression problem, the output of the neuron in the output layer should be a real number. This can be achieved by using a linear activation function, see Table 7.5, which simplifies the estimation of the artificial neuron, Equation (7.11), into the estimation of the Linear Regression, Equation (7.1).

For the hidden layers, many activation functions can be used. In general, non-linear activation functions are used to enable the model to learn non-linear relations. See Table 7.5 and Figure 7.8 for an overview of common activation functions [46].

Activation function	**Formula**	**Output range**
linear	$\psi(x) = x$	$(-\infty, \infty)$
threshold function	$\psi(x) = \begin{cases} 1, \text{if } x \geq 0 \\ 0, \text{if } x < 0 \end{cases}$	$\{0, 1\}$
hyperbolic tangent	$\psi(x) = tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$	$(-1, 1)$
logistic	$\psi(x) = \sigma(x) = \frac{1}{1+e^{-x}}$	$(0, 1)$
rectified linear unit	$\psi(x) = max(0, x)$	$[0, \infty)$

TABLE 7.5: Activation functions [46].

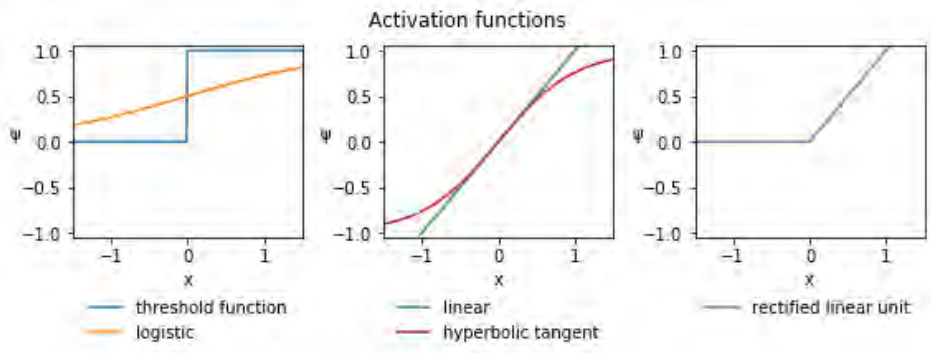


FIGURE 7.8: Plot of the activation functions.

In the recent years, the Rectified Linear Unit (ReLU) [47] is most commonly used as activation function in the hidden layers [48]–[51] because of its performance and speed. Therefore, ReLU is selected as activation function in the hidden layers in this research.

7.8.1 Optimization algorithms

During the training phase of the model, the weights w and bias b in each neuron should be optimized. For this purpose, the gradient could again be used for the direction of the optimization steps (Section 7.5) [51], for which we should specify an objective function. An example of an objective function is the least squares, as was used in Linear Regression (Section 7.1). Another objective is the minimization of the mean absolute error (MAE), see Equation (7.12), in which y_i is the actual observation and $\hat{y}_i$ the estimation of sample i. The

mean absolute error equally weights the errors, in contrast to the mean square error (MSE), which is the average of the squared residuals, which gives bigger weights to larger error, as mentioned in Section 7.2. Since all errors deserve similar attention in the estimation of the production durations, the MAE is used as objective function. For example, using MAE, a model which makes two times an error of 5 seconds, or a model which makes one time an error of 1 second and one time an error of 9 seconds are considered equally good. Since all errors are equally important for the MAE method, it is more robust to outliers than the MSE method. Since the used outlier detection method does not guarantee that all strange values are filtered out, a robust method is preferred.

$$\begin{aligned} &\text{Objective} = min\ L \\ &L = \frac{1}{n}\sum_{i=1}^{n}|r_i| \\ &r_i = y_i - \hat{y}_i \end{aligned} \tag{7.12}$$

Since the non-linearity of Neural Networks (as for GBM in Section 7.5) "causes most interesting loss functions to become nonconvex" [51], multiple minima could be found, see Figure 7.9. For this type of problems, *Gradient Descent* (or *Steepest Descent*) is commonly used to optimize the parameters θ (the weights of a Neural Network) to receive the optimum value of the objective function L, using an iterative process in which the update in each iteration is in the direction of the negative gradient g [51], see Equation (7.13), as mentioned in Section 7.5. The *learning rate* ϵ determines the step size in the direction of the gradient, which should be big enough for significant steps towards a minimum, but small enough for convergence to an acceptable minimum, see Figure 7.9. In such iteration, commonly called *epoch*, all weights (θ) in the network are updated using the objective function (L) over all training samples. The number of epochs should be large enough for convergence towards a minimum, but small enough to compute.

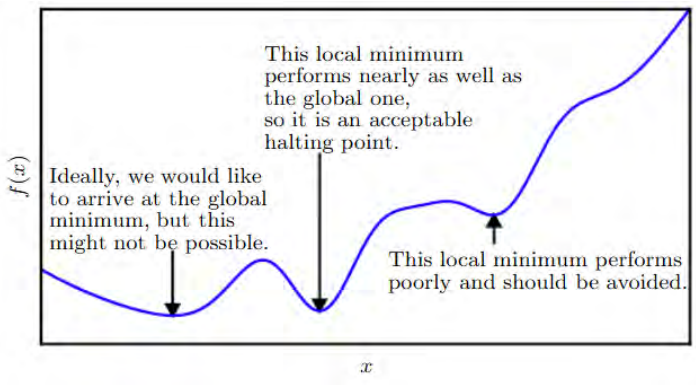


FIGURE 7.9: Example of nonconvex objective function [51].

$$\begin{aligned} &g = \nabla_\theta L(\theta) \\ &\theta \leftarrow \theta - \epsilon g \end{aligned} \tag{7.13}$$

The disadvantage of Gradient Descent (GD) is that it becomes very slow when the size of the training set is very large, because the computational costs of a single gradient step becomes prohibitively long [51]. To limit the computational costs, Stochastic Gradient Descent (SGD) was introduced, which uses the insight that the objective function is usually created

per sample and averaged over the whole training set [51]; see our objective function in Equation (7.12). The gradient could be approximated by calculating the gradient $\tilde{g}$ over smaller number of samples (B), called *batch*, see Equation (7.15). In SGD the number of batches D in one epoch is equal to the total sample size (n) divided by the batch size (B). Hence, instead of a single weight update per epoch (GD), the weights (θ) are updated D times per epoch (SGD), which requires a smaller number of epochs for convergence. When a batch contains only a single sample, the stochastic approximation of the gradient can be very different over the batches, resulting in a "zig-zag" path towards a minimum.

$$L = \frac{1}{n} \sum_{i=1}^{n} L_i \tag{7.14}$$

$$\begin{aligned} \tilde{g} &= \frac{1}{B} \sum_{i=1}^{B} \nabla_\theta L_i(\theta) \\ \theta &\leftarrow \theta - \epsilon \tilde{g} \end{aligned} \tag{7.15}$$

Usually, batch sizes are small; ranging from one to a few hundreds and "especially when using GPUs, it is common for power of 2 batch sizes to offer better runtime" [51]. The recent study of Masters and Luschi (2018) showed that a batch size between 2 and 32 is recommended [52].

In addition, other methods became available. Where SGD sometimes was still considered as slow, a new method Momentum [53] was introduced. This method tries to decrease the "zig-zag" effect of SGD by remembering the previous direction and updating it to the current gradient with a certain weight. In this way, oscillations of the gradient are damped. Therefore, the step size depends on "how large and how aligned a sequence of gradients are" [51]. Figure 7.10 shows an example of Momentum.

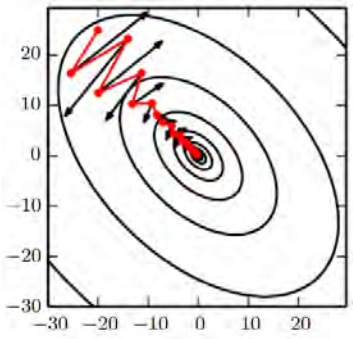


FIGURE 7.10: Example objective function with optimum at (0,0). The black arrows indicate the direction of the gradient of the objective function, the red lines show the path taken by Momentum [51].

In 2011, the Adaptive Gradient Algorithm (AdaGrad) [54] was introduced, which has instead of one learning rate ϵ, a learning rate for each parameter. This allows the method to "dynamically incorporate knowledge of the geometry of the data observed in earlier iterations to perform more informative gradient-based learning" [54]. In general, AdaGrad outperforms SGD with Momentum if the gradients are *sparse* (mostly zero) [55].

In addition, Root Mean Square Propagation (RMSProp) is another method with per-parameter learning rates, which "modifies AdaGrad to perform better in the nonconvex

setting by changing the gradient accumulation into an exponentially weighted moving average", which comes with the introduction of the *decay rate* ρ that controls the period of moving average on past squared gradients [51], [56], [57]. Using this parameter, RMSProp is able to forget previous iterations, in contrast to AdaGrad in which the learning rate depends on the entire history [51]. Therefore, RMSProp works well in "non-stationary settings" [55], for example, due to noise in the data.

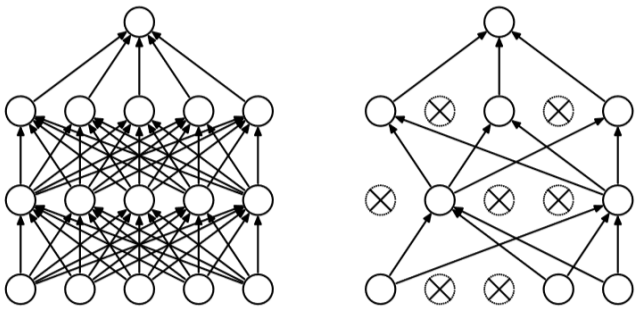
In 2014, Kingma and Ba introduced another method called *Adam*, which is derived from "adaptive moment estimation" [55]. Adam combines RMSProp and Momentum (Nesterov's variant [58]): besides the decay rate on the squared gradients (similar to RMSProp), it applies the same approach on the past gradients (similar to Momentum) [57]. To account for the initial values of the gradient estimations, Adam introduces *initialization bias correction*. For the complete algorithm, please refer to the work of Kingma and Ba (2014) [55].

According to Kingma and Ba (2014), Adam has multiple advantages: "the method is straightforward to implement, is computationally efficient, has little memory requirements, is invariant to diagonal rescaling of the gradients, and is well suited for problems that are large in terms of data and/or parameters." [55]. On top of that, Adam is "fairly robust to the choice of hyperparameters" [51], [55]. Finally, S. Ruder (2016) provides an overview of modern optimization algorithms and recommends to use Adam because of its *initialization bias correction* [57].

Since an ANN consists of multiple artificial neurons, the prediction can be obtained by applying Equation (7.11) recursively. To obtain the gradient of the objective of the first neuron, the *chain rule of calculus* can be used. Hence, for the objective function of each neuron, the chain rule could recursively be applied. Note that because of the chain rule, the gradients of the neurons are very similar. Therefore, we can store intermediate results to save computations. This procedure is commonly referred to with *Backpropagation* [51]: the error in the iteration is "propagated backwards" through the network. In this way, the gradient of each batch can be obtained and be used for the weight updates.

7.8.2 Dropout

Since combining many artificial neurons leads to many parameters / weights that should be optimized, the neural network is prone for over-fitting. Many methods exist to reduce model complexity. In 2014, Google introduced such a method, called *dropout* [59], [60]. This method randomly "drops" neurons from the network, see Figure 7.11 for a visualization. The idea behind temporarily disabling neurons is to prevent units "from co-adapting too much" [59].



(a) Standard Neural Net (b) After applying dropout.

FIGURE 7.11: Abstract visualization of a feed-forward Neural Network before applying dropout (a) and after applying dropout (b) [59].

During the training phase, each node in a layer is retained with probability p and is dropped with probability $1 - p$. After training, during the test phase, all neurons will be present and the weights are multiplied with p. "This ensures that for any hidden unit the expected output (under the distribution used to drop units at training time) is the same as the actual output at test time" [59]. Srivastava et al. (2014) found that a Neural Network with dropout and using the approximate averaging method at test time "leads to significantly lower generalization error on a wide variety of classification problems compared to training with other regularization methods" [59].

7.8.3 Hyperparameters

An ANN could be used for the estimation of the production durations. In general, deep Neural Networks are used for complicated classification tasks, like, image recognition [61], [62] and speech recognition [63]. Deep Neural Networks are not frequently used in regression problems. Therefore, we decided to use a shallow Neural Network of an input layer, one or two hidden layers and an output layer. As mentioned before, a ReLU activation function is used in the hidden layers and a linear activation function in the output layer.

The values 2, 4 and 8 are tested for the number of neurons in each hidden layer. In addition, to analyse if the model benefits from the introduction of dropout, dropping probabilities 0 (no dropout) and 0.2 are both evaluated on the input layer and the hidden layer(s), of which the latter is a typical value for dropout according to Srivastava et al. [59]. An overview of the tuned hyperparameters is given in Table 7.6.

Hyperparameter	**Description**	**Evaluated values**
batch_size	Number of samples in one batch	16
epochs	Number of epochs	Use early stopping on 30% validation data, with maximal 100 epochs
n_layer1	Number of nodes in hidden layer 1	2, 4, 8
n_layer2	Number of nodes in hidden layer 2	0, 2, 4, 8
dropout_input_layer	Dropout rate on visible layer (input layer)	0, 0.2
dropout_hidden_layer	Dropout rate on hidden layer(s)	0, 0.2

TABLE 7.6: Hyperparameter settings of the Neural Network.

For the optimization of the weights of the Neural Network, Adam optimizer will be used, which was recommended by S. Ruder (2016) [57]. The default hyperparameters of Adam were used in this research, because the method is fairly robust to the choice of them [51], [55]. For this iterative optimization procedure, we need to specify the number of epochs and the batch size. The recent study of Masters and Luschi (2018) recommends a batch size between 2 and 32 [52]. Since it is common to use a power of 2 (i.e. 2, 4, 8, 16, 32, ...), we decided to choose a batch size of 16.

The number of epochs that is needed for convergence is problem dependent. To evaluate if the model is under-fitting or over-fitting the training data, a validation set can be used to which the performance can be compared. When the model both improves on the training and validation set, it indicates that the optimizer did not converge so far. When the model improves on the training set but does not on the validation set (or even get worse), it indicates that the model starts over-fitting the training set. A technique which can be used to stop training at the 'right' moment is *early stopping*. Using this technique, the optimizer stops when the performance on the validation set does not improve any more. According

to S. Ruder (2016), early stopping should always be used while training an Neural Network [57]. In this research, it was chosen to use 70% of the training data to fit the Neural Network weights, and 30% of the training data as validation data to evaluate the performance. Besides, to save computations, the training is stopped as soon as possible, without patience.

Finally, to efficiently perform Backpropagation, all input signals (features) were normalized before passing to the network [64]. A standard scaler is used for this, which subtracts the mean and scales to unit variance.

7.9 Evaluation methods

In the description of the machine learning models was mentioned that some evaluation will be performed to choose the right hyperparameters. However, the used procedure was not explained in detail yet, which we will do now.

7.9.1 Training and test set

As mentioned, twelve different target variables are aimed to be estimated in this research. Therefore, twelve final models are created: one for each target. The data samples that are used to create a model consist of the batches that were produced and of which the target duration is observed. In this way, the model estimation could be compared with the observed production duration. However, it is important to conclude the model performance on a different sample set as used for the creation of the model, due to the possibility of over-fitting. Therefore, the data samples are split in a *training set* and a *test set*.

Most of the benchmark models (Table 2.1) are time-based: only the last x batches are used for the estimation. Therefore, we need to split the training and test set in such a way that this time property is not affected. This is ensured by splitting the training and test set in time. It was chosen to use the last three months of 2018 (2018/10/01 – 2018/12/17) as test set, and all records before 1 October 2018 as training set (2016/08/14 – 2017/08/14 and 2017/12/14 – 2018/09/30). The test set consists of approximately 10% of the data samples using this approach. This results in the following amounts, see Table 7.7.

Note that the number of batches are different over the targets due to the outlier filtering performed in Section 4.3. Also, the first few warm-up periods of the pressing durations could not be estimated because at least a few observations are needed to perform the decomposition using the Robust Regression model in a windowed fashion. This results in a few missing speed target values in the training set. These samples are simply ignored.

7.9.2 Prediction horizon

The most recent batches are assumed to be most representative for the current production durations, due to new harvests of ingredients, newly created articles and seasonal effects. Therefore, when the final model proposed in this research will be used in practice, it will be retrained frequently. Since the most computational effort of the selected machine learning models is at training time, it would be reasonable to train the machine learning models during the night, to limit the occupation of computing power during the day. With this in mind, it is decided to use a prediction horizon of one day. Since there are 78 days in the test set (31 in October, 30 in November and 17 in December), the models will be trained 78 times and be evaluated on the next day. Therefore, the training set as defined in Section 7.9.1 is incrementally enlarged with one day at the time. This is visualized in Figure 7.12.

Note that the selection of 78 days results in many more evaluated production durations because many batches are produced every day, see Table 7.7 for the number of test samples.

Target	Training samples	Test samples
HA1_TOEV	106,567	13,960
HA1_ZEEF	38,683	4,780
BU6_LOS	110,090	14,232
NM1_step	103,748	13,279
NM1_Los_step	111,267	14,397
MML_AFV_TR	109,292	14,221
MML_AFV_TR_NADRAAI	59,791	7,701
PL_PO1_slope	12,866 (+32*)	1,712
PL_PO2_slope	11,518 (+8*)	1,530
PL_PO3_slope	9,931 (+17*)	1,388
KO3_idle_time	10,172	1,327
PL_PO4_slope	13,835 (+13*)	1,865

TABLE 7.7: Number of samples in the training and test sets for the estimation of the twelve targets. * = missing target observations due to decomposition of period A and B at press line.

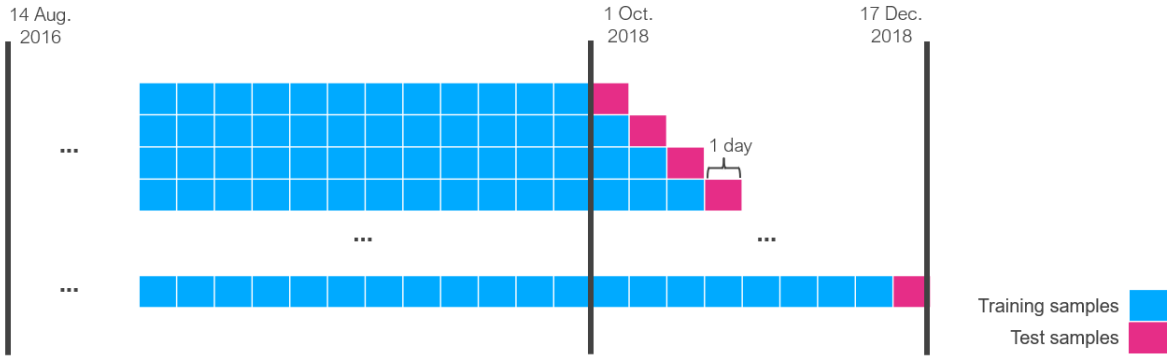


FIGURE 7.12: Evaluation approach on test set using a planning horizon of 1 day. One row represents the training of the model, in which the blue samples are used for training and the pink samples are used for testing.

7.9.3 Goodness of fit

As mentioned, the estimation of the model will be evaluated against the observed production duration. This comparison will always be made between the total estimated duration and the total observed duration. Hence, the pressing duration decomposition will be reversed before evaluation.

Several goodness of fit measures exist. For example, the Mean Absolute Error (MAE) and the Mean Square Error (MSE) were already mentioned in Section 7.8.1. In that section, it was mentioned that all errors on the production durations (estimated versus observed) deserve similar attention, which resulted in the choice of the MAE. MAE has more advantages compared to other goodness of fit measures. For example, it is highly interpretive because it explicitly measures (in seconds) the performance of the model. Therefore, the MAE will be used to determine which model is the best.

Also, the Mean Absolute Percentage Error (MAPE) is calculated for each model. This metric computes the percentage error of each batch and takes the average value of these. The percentage error compares the absolute error to the production duration. Therefore, similar errors result in a big percentage error when the production duration is small and in a small percentage error when the production duration is big.

7.9.4 Missing values

Missing values may occur in the features of the training samples. For example, when the BOA is not used at press line 2, the BOA opening size will be missing. Besides, the article features are created by looking at the past observations of the same article. However, when a new article is introduced, these observations are missing.

The missing values can be split in setpoint related features (described in Section 5.3) and other features. Missing values in the setpoint related features occur only when the value does not exist. Therefore, these missing values are imputed with zero. In contrast, the missing values of the other features occur when the value is unknown. Therefore, these missing values are imputed with the median value of the feature in the training set, because the median value is robust for outliers.

7.9.5 Grid search for hyperparameter and feature selection

Previously, many features and many hyperparameters were discussed. To choose which should be used in the final model, a *grid search* is performed. A *grid* can be created by selecting all possible combinations of the features and hyperparameters. A single *grid point* represents a single combination. The idea is to evaluate all grid points and search for the best grid point, which represents the best selection of hyperparameters and features for the model.

The evaluation of the grid points can be performed in the same way as the final model comparison: split the data in a training and test set, and use a similar evaluation approach as visualized in Figure 7.12. To avoid confusion, the training set used in the grid search will be called *exploration set*, and the test set will be called *validation set*. The validation set is chosen to be the last two months of the training set: August and September 2018, and the exploration set will contain all months before August 2018. Splitting the data in this way, the validation set contains again approximately 10% of the samples. Note that the grid search will only use training samples. The test samples will never be used for training.

The validation set consists of 61 days (31 days in August and 30 days in September). Training and testing a model 61 times for a single grid point results in a lot of computations. Therefore, only a random selection of days will be used. Since it is important to evaluate different hyperparameters (they affect the model performance), it is important to reduce the computation time of a single grid point. To keep the average duration of a single grid point of the tree-based models (these models have the most hyperparameters) below 60 seconds, maximal four days could be evaluated. Therefore, four days were randomly selected from the validation set. For fair comparison, the same four days will be used for all models. Note that the selection of four days results in many more evaluated production durations because on average 170 GML batches and 19 batches per PL are produced per day.

The evaluation on the four validation days is done in the same way as before, using the MAE. The grid point with the smallest MAE is selected as the best hyperparameter and feature combination of that model. The resulting best versions of the models will be compared on the test set, which was described earlier.

Feature sets

Many features were described in Section 5. To limit the created number of grid points, and thus the number of evaluations, the features are split into feature sets, see Table 7.8. Of course, this choice creates some noise because a single feature set could contain both useful and useless features. However, it gives an idea of how the different kind of features contribute to the model performance.

Feature set	Features
1	batch size and article features
2	batch size, article and setpoint features
3	batch size, article, setpoint and ingredient features
4	batch size, article, setpoint, ingredient and weather features
5	batch size, article, setpoint, ingredient, weather and time features
6	batch size, article, setpoint, ingredient count, ingredient group, weather and time features
7	batch size, setpoint, ingredient, weather and time features

TABLE 7.8: Feature sets.

The first feature set contains the batch size (Section 5.1) and article features (Section 5.6), because these features show in general the most correlation with the production durations, see Figures 6.4, 6.7, 6.11B, 6.14A, 6.20B and 6.21B. The second feature set includes the features from the first feature set, and on top of that the setpoint features (Section 5.3). The setpoint features consist of features like the sieve size, the supply capacity and which press units are used, which are highly correlated with the production durations, see Figures 6.2B, 6.18A, 6.21A, 6.23B and 6.25A.

The third feature set also contains the ingredient features (Section 5.4), which consist of ingredient percentage features and an ingredient count feature. These features are used on top of the other features to see if the models can relate some variation in the production durations to the ingredient composition of the batch.

The fourth feature set also includes the weather features, which show little correlation with the production durations according to the feature analyse performed in this research, see Section 6. The fifth feature set contains in addition the time features, which also show little correlation with the production durations according to the feature analysis. Note that the time-correlated production durations were concluded to be estimated by a simple time-based model (Section 6).

The sixth feature set is included to see the effect of summarizing the different ingredients to their ingredient group. The ingredient percentage features are removed from this feature set. Finally, the seventh feature set is included to get an idea of the model performance when the article information cannot be used at all.

Simple time-based models

The evaluated grid points for the advanced machine learning models are described in the first part of this section for each explained model. For the models for which in Section 6 was concluded to fit a simple time-based model to the last B batches, are the following numbers evaluated in the grid search: 1, 3, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 200, 300, 400, 500, 600, 700, 800, 900 and 1000. Note that the simple time-based models do not use the different feature sets. Therefore, the grid of the simple time-based models is much smaller (and the grid search much faster) than the grid of the advanced machine learning models. Therefore, the limitation of four days is not needed. The grid search of the simple time-based models is evaluated against all days in the validation set.

7.9.6 Evaluation final results

Previously was described how the performance of the models is measured. These models can be compared using this performance measure. The final results will be compared in three ways.

First of all, the model performances will be compared to the performance of the benchmark, see Table 2.1. This is done by filtering on all test samples for which the benchmark could give an estimation.

The advantage of the benchmark model is that it is very fast to compute. This makes it possible to use all data points right before the estimation. However to create a fair comparison, two benchmarks are created. Benchmark 1 uses all produced batches prior to the estimated batch, which is compared with Benchmark 2, which uses only the batches of the previous day and earlier. The used data by Benchmark 2 is similar to the data used by the proposed machine learning models in this research, which are trained during the night. Both benchmarks provide information about the predictive power. When a schedule is created for the next day, Benchmark 2 provides the performance of the production duration estimations. In contrast, when an estimation should be given for then next batch, Benchmark 1 provides the benchmark performance. The benchmark results will be compared to evaluate the importance of the last batches to the estimation of the production duration.

Secondly, the performance of the models on the samples for which the benchmark (Benchmark 2) could not give an estimation (e.g., new products) are compared to the performances with benchmark results. This shows how the model performs in the situations that the benchmark could not give an estimation. For the target durations which were estimated by a median per article, this comparison gives an idea of how the model performs on new products. Table 7.9 shows the number of missing values in the predictions of Benchmark 2 on the test set.

Finally, all final models are evaluated on the test set while training on a smaller training set, which only contains the last x batches. This evaluation provides information on how many training data should be available at the animal feed plants before the model performs well. For the value of x, we select 1, 2, 6 and 12 times the median number of batches per month, see Table 7.9. Therefore, we can conclude if respectively 1, 2, 6 or 12 months of training data are desired for the use of the model.

Target	**Median number of batches per month**	**Number of missing values by benchmark on test set**
HA1_TOEV	4,877	155
HA1_ZEEF	1,769	0
BU6_LOS	5,139	59
NM1_step	4,858	517
NM1_Los_step	5,200	0
MML_AFV_TR	5,059	60
MML_AFV_TR_NADRAAI	2,788	0
PL_PO1_slope	590	216
PL_PO2_slope	534	84
PL_PO3_slope	454	70
KO3_idle_time	469	0
PL_PO4_slope	628	104

TABLE 7.9: Median number of batches per month of each target variable and the number of missing values estimated by the benchmark.

7.9.7 Feature importance

After comparing the final models as mentioned above. The best model is chosen for which the importance of the features is analysed. For the linear and robust regression models, this is done by analysing the fitted weights of the model. For the tree-based models Extra Trees and LightGBM is the feature importance relatively easy to obtain. Each split in the Decision Trees uses one feature. The feature importance can be derived from the improvement that each split has made on the objective function. Features which significantly improve the predictive power of the model are valued as the most important features.

The feature importances of the Neural Network are less straight-forward to obtain. One way to achieve this is to use a shuffling technique [65]–[67]. The trained Neural Network will be evaluated on an adapted test set. In this adapted test set, the values of a certain feature, say feature a, are shuffled over the different batches. The relationship between the target production duration and feature a is disturbed by this shuffle. Therefore, we expect to see an increase in the Mean Absolute Error (MAE) of the model on the adapted test set. How much the MAE increases indicates the importance of feature a. If feature a is not important at all, the model did not find a strong relationship between feature a and the production duration, so the shuffling will not have a big impact on the performance. In contrast, if feature a is very important, the shuffling disturbs the strong relationship between feature a and the production duration, and therefore a significant decrease of the model performance (an increase of MAE) is expected. This procedure will be applied to all features, one at the time, which results in an estimation of the feature importances in the Neural Network.

7.9.8 Implementation

All analysis and modelling is implemented in Python™, which is a common programming language in data science. The package Scikit-learn[1] is used for the implementation of Linear Regression and Extra Trees, Statsmodels[2] for the implementation of Robust Regression, and the packages LightGBM[3] and Keras[4] for the implementation of respectively LightGBM and Neural Networks. All computations are performed on a single Windows 10 laptop with the following specifications: 2,20 GHz Intel® Core™ i7-8750H with 16GB RAM.

[1] https://scikit-learn.org/stable/index.html
[2] https://www.statsmodels.org
[3] https://lightgbm.readthedocs.io
[4] https://keras.io/

8 Results

In this section, the results are given for each target variable (Table 2.1). This happens in the following way. First, the results of the grid search are given, as explained in Section 7, in order to obtain the final models. Secondly, the benchmark results are given and the best model is selected based on the three evaluation methods described in Section 7.9.6. Finally, the importance of the features in the best model is analysed.

8.1 HA1_TOEV

The first target variable is HA1_TOEV, which is the duration of the grinding step in the GML. First, the grid search is applied to find the best hyperparameters and feature set of the five machine learning models; Linear Regression, Robust Regression, Extra Trees, LightGBM and Neural Network. Since the HA1_TOEV durations show a strong linear relationship with the batch size, see Figure 4.6A, the grid search is performed twice: once to predict the total duration and once to predict the duration per 1,000 kg (and multiply the batch size afterwards). For all models, the best target is found to be the duration per 1,000 kg. The best feature set and hyperparameters of each model are shown in Table 8.1.

Model	**Feature set**	**Hyperparameters**	**MAE**	**Std**
Linear Regression	6	fit_intercept = True	19.092	5.145
Robust Regression	3		18.106	4.970
Extra Trees	3	max_depth = 30 min_samples_leaf = 10	18.181	7.271
LightGBM	3	max_depth = 8 min_data_in_leaf = 10 num_leaves = 50 n_estimators = 300 learning_rate = 0.1 lambda_l2 = 1	16.806	5.487
Neural Network	1	n_layer1 = 8 n_layer2 = 0 dropout_input_layer = 0 dropout_hidden_layer = 0	17.541	3.744

TABLE 8.1: Grid search results for the estimation of the HA1_TOEV durations.

To observe how much spread is in the created models of one grid point, the standard deviation (std) is calculated over the MAE of the individual trained models, in other words, the MAE of the four days in the validation set. Note that the same four days are used for all models and all grid points for a fair comparison. Spread in the performance over the four days can be observed, see Table 8.1. According to Table 8.1, the best grid point of the Neural Network has the smallest spread over the four days compared to the best grid point of the other models.

The final models are obtained by selecting the feature set and hyperparameters of the best grid point, see Table 8.1. These final models will be compared to the benchmark and to each other on the test set. There are 78 days in the test set. The standard deviation (std) will be calculated over the individual daily MAEs as a measure of the spread in the performance of the model.

First, the benchmark model will be evaluated. According to Table 8.2, the MAE of Benchmark 2 is approximately 3 seconds (13%) higher than the MAE of Benchmark 1, which is found to be significant while using the t-test (p-value of t-test « 0.01). Therefore, the most recent batches (of the same article) provide a significant predictive power to the benchmark model.

Model	**MAE**	**MAPE**	**Std**	**Number of predictions**	**Number of unknown predictions**
Benchmark 1	17.879	6.620	6.242	13,820	140 (1.003%)
Benchmark 2	20.289	7.445	6.451	13,805	155 (1.110%)

TABLE 8.2: Benchmark results for the estimation of the HA1_TOEV durations.

Table 8.3 shows the results of the final models on the test set, including the split to batches with and without benchmark estimations. The results show that all evaluated models outperform the benchmark, which is the answer to the second sub-question of this research. According to the final results, the Neural Network is the best model with an MAE of 16.293. However, the model LightGBM shows better estimations for the batches without benchmark estimations, see Table 8.3.

	With benchmark estimation		**Without benchmark estimation**		**All**		
Model	**MAE**	**MAPE**	**MAE**	**MAPE**	**MAE**	**MAPE**	**Std**
Linear Regression	17.495	6.529	26.539	9.593	17.595	6.563	5.848
Robust Regression	16.371	5.935	23.961	8.678	16.455	5.965	5.584
Extra Trees	16.853	6.354	22.000	7.861	16.910	6.371	5.565
LightGBM	16.423	6.153	**20.436**	7.448	16.467	6.167	5.475
Neural Network	**16.182**	5.914	26.174	9.376	**16.293**	5.952	5.519
Benchmark 2	*20.289*	*7.445*					

TABLE 8.3: Model results for the estimation of the HA1_TOEV durations.

The difference between the results of LightGBM and the Neural Network can also be observed from their usage of the features. The final LightGBM model uses feature set 3, which includes batch size, article, setpoint and ingredient features, and the final Neural Network model uses feature set 1, which only includes the batch size and article features. The feature importance of the two models is shown in Figures 8.1 and 8.2.

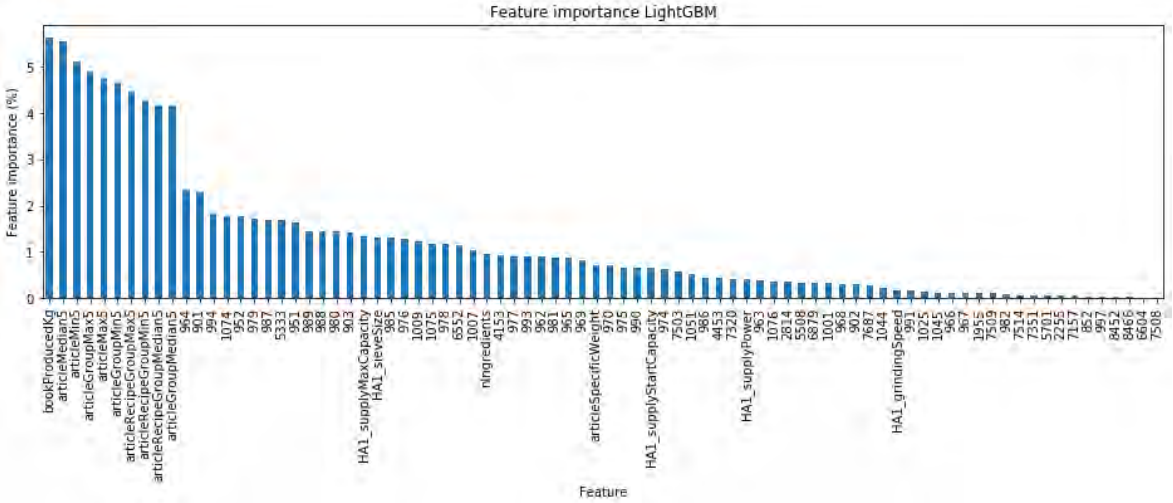


FIGURE 8.1: Feature importance LightGBM for the estimation of the HA1_TOEV durations. (19 features are not shown, because these are not used by the model.)

According to Figure 8.1, not only the batch size and article features are important to the LightGBM model, but also some ingredients and setpoint features. For example, the ingredients *964*, *901* and *994* show importance in the predictions of LightGBM, see Figure 8.1.

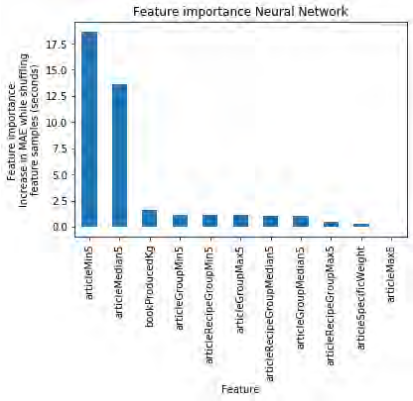


FIGURE 8.2: Feature importance Neural Network for the estimation of the HA1_TOEV durations.

In contrast to LightGBM, the Neural Network estimations mostly depend on the features *articleMin5* and *articleMedian5*, and the importance of the batch size (*bookProducedKg*) feature is much smaller, see Figure 8.2. Hence, the Neural Network works better when already some batches of an article are observed (*articleMin5* and *articleMedian5* are known / not imputed with the total median duration), which is the case for the batches where the benchmark estimation exists too. This causes the difference in performance between the LightGBM and the Neural Network model on the batches with and without benchmark estimation, see Table 8.3.

Promising results are likely to be implemented by ENGIE Industrial Automation in other industrial plants too. Therefore, we trained the model on fewer data samples as an indication of how much training data is needed. The results are shown in Table 8.4. Table 8.4 shows that the Robust Regression model performs best on fewer records compared to the other suggested models.

	1 month			2 months		
Model	**MAE**	**MAPE**	**Std**	**MAE**	**MAPE**	**Std**
Linear Regression	20.767	7.922	10.836	18.105	6.790	6.330
Robust Regression	**16.128**	5.756	5.577	**16.018**	5.716	5.688
Extra Trees	16.985	6.271	5.474	16.508	6.091	5.575
LightGBM	18.547	6.886	5.470	17.839	6.649	5.766
Neural Network	17.564	6.410	5.802	16.932	6.122	6.008

	6 months			12 months		
Model	**MAE**	**MAPE**	**Std**	**MAE**	**MAPE**	**Std**
Linear Regression	17.675	6.609	5.982	17.660	6.608	5.945
Robust Regression	**16.390**	5.912	5.738	**16.527**	5.976	5.574
Extra Trees	16.664	6.207	5.629	16.906	6.343	5.541
LightGBM	16.987	6.343	5.641	16.709	6.254	5.318
Neural Network	16.853	6.188	5.744	16.826	6.235	5.692

TABLE 8.4: Model results with smaller training sets for the estimation of the HA1_TOEV durations.

The Robust Regression model uses the same feature set as the LightGBM model. In contrast to the feature importance of LightGBM, the feature coefficients of the Robust Regression function show directly if the feature positively or negatively influences the production duration.

Figure 8.3 shows two features with a coefficient ≤ -3 and five features with a coefficient ≥ 3. These features are ingredient features and happen to be non-zero for only one or two batches. Therefore, these seven coefficients are likely the results of over-fitting to those few observations. However, the model performs well (low MAE compared to the benchmark) and is able to achieve this performance with only one month of training data.

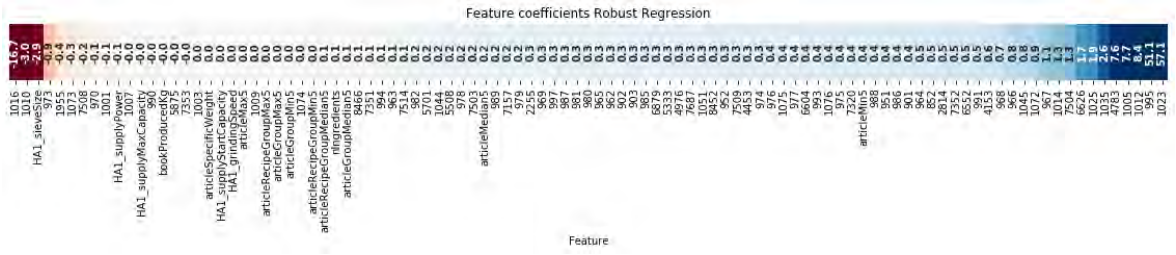


FIGURE 8.3: Feature coefficients of the Robust Regression model for the estimation of the HA1_TOEV durations.

The first sub-question of this research is "what are the explanatory variables for the production duration?". According to Figures 8.1, 8.2 and 8.3, the most important features for the estimation of the HA1_TOEV duration are the *sieve size*, *articleMin5* and *articleMedian5* features. However, the importance of the features depend on which model is used.

The estimations of the benchmark model and the Robust Regression can be observed in Figure 8.4 for the observations in the test set. The diagonal line indicates the situation of a perfect prediction (estimation=observation). Figure 8.4 shows indeed that the estimations of the Robust Regression are closer to the diagonal than the estimation of the benchmark model.

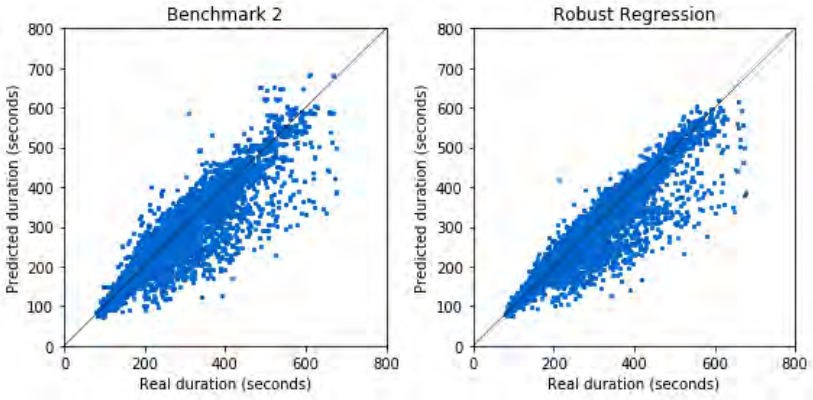


FIGURE 8.4: Observed HA1_TOEV durations versus estimated HA1_TOEV durations by Benchmark 2 and Robust Regression.

According to both the results based on the full training set as on a smaller training set, the Robust Regression model is the best choice for the estimation of the HA1_TOEV durations. This model reduces the MAE from 20.289 (Benchmark 2) to 16.371 for the same batches and is able to give an estimation for all future batches.

8.2 HA1_ZEEF

The estimation of the HA1_ZEEF durations is done by a simple time-based model, as explained in Section 6. A grid search was performed to obtain the number of batches per sieve switch which are used for the estimation. The optimal number of batches in this grid is found to be 70 batches, resulting in an MAE of 0.988 seconds. A final model is created, which estimates the production durations by calculating the median duration of the past 70 batches that perform the same sieve switch. This model is tested on the test set and compared with the benchmark.

Benchmarks 1 and 2 are equal for the sieve switching durations. This happens because the median duration since the first of January 2017 keeps the same for all estimations in the test set (73 seconds). Therefore, the median at night (Benchmark 2) is the same as during the day (Benchmark 1). The benchmark makes on average an absolute error of only 1.642 seconds, see Table 8.5.

Model	**MAE**	**MAPE**	**Std**	**Number of predictions**	**Number of unknown predictions**
Benchmark 1 and 2	1.642	2.187	0.504	4780	0 (0%)

TABLE 8.5: Benchmark results for the estimation of the HA1_ZEEF durations.

Running the final model on the test set gives an MAE of only 0.864 seconds, see Table 8.6, which is approximately half of the MAE of the benchmark. In addition, the standard deviation of the MAE over the 78 days in the test set is 0.285 seconds, which is also smaller than for the benchmark (0.504 seconds).

Model	**MAE**	**MAPE**	**Std**
Median duration of the last 70 batches with the same sieve switch	**0.864**	1.158	0.285
Benchmark 2	*1.642*	*2.187*	*0.504*

TABLE 8.6: Proposed model results for the estimation of the HA1_ZEEF durations.

Figure 8.5 shows the estimations of the benchmark and the proposed model for the observations in the test set. Note that the estimated duration of the benchmark does not change for the batches in the test set, which results in a horizontal line in Figure 8.5. Figure 8.5 shows a better fit to the diagonal line for the proposed model than for the benchmark, which results in the lower MAE.

The t-test can be used to determine if the difference between the proposed model and the benchmark is significant. The null hypothesis of the t-test is that the means of the absolute errors are the same. The p-values is found to be much smaller than 0.01. Therefore, we can conclude that our proposed model for the estimation of the HA1_ZEEF durations is better than the benchmark.

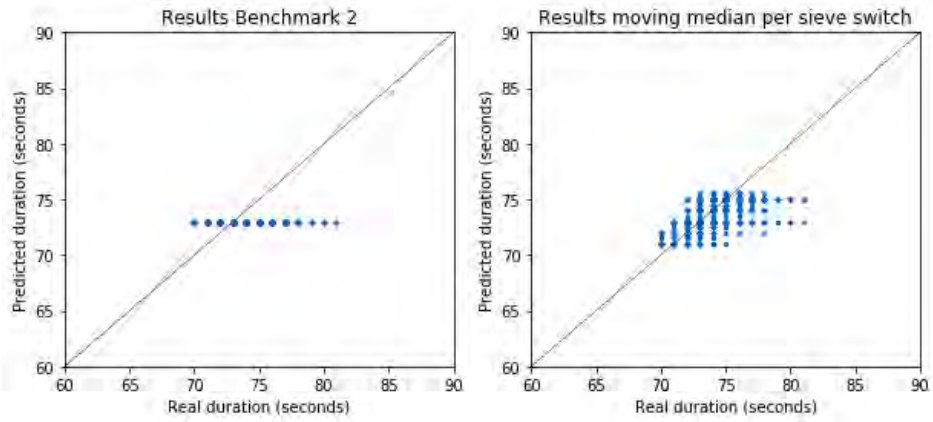


FIGURE 8.5: Estimated durations versus observed durations for the benchmark (left) and proposed model (right) for the estimation of the HA1_ZEEF durations.

8.3 BU6_LOS

The estimation of the BU6_LOS durations is done by a simple time-based model too, as explained in Section 6. Again, a grid search was performed to obtain the number of batches which are used for the estimation. The optimal number of batches in this grid is found to be 50, resulting in an MAE of 2.376 seconds on the validation set. The final model estimates the durations by calculating the median duration of the past 50 batches. This model is compared to the benchmark on the test set.

Benchmark 1 and 2 have respectively an MAE of 5.001 and 5.020 seconds on the test set, see Table 8.7. This indicates that the most recent batches are not very important for the predictive power of the benchmark, this is because the durations are fairly constant, see Figure 4.12 in Section 4.

Model	**MAE**	**MAPE**	**Std**	**Number of predictions**	**Number of unknown predictions**
Benchmark 1	5.001	11.796	2.250	14,200	32 (0.225%)
Benchmark 2	5.020	11.844	2.254	14,173	59 (0.415%)

TABLE 8.7: Benchmark results for the estimation of the BU6_LOS durations.

The disadvantage of the benchmark is that it is not able to give an estimation for new articles. Therefore, less than 0.5 percentage of the batches could not be estimated by the benchmark, see Table 8.7.

The proposed model gives an estimation based on the most recent 50 batches. It is, therefore, able to estimate all batches in the test set. In addition, the MAE of the final model

is smaller than the MAE of the benchmark for both batches with (3.919 seconds) and without benchmark (4.686 seconds), see Table 8.8.

Again, the t-test can be used to determine if the proposed model is significantly better than the benchmark. The proposed model is found to be significantly better than the benchmark because the p-value of the t-test is much smaller than 0.01. Therefore, we can conclude that the proposed model performs significantly better than the benchmark.

	With benchmark estimation		**Without benchmark estimation**		**All**		
Model	**MAE**	**MAPE**	**MAE**	**MAPE**	**MAE**	**MAPE**	**Std**
Median duration of the last 50 batches	**3.919**	9.133	**4.686**	10.634	**3.923**	9.139	2.283
Benchmark 2	*5.020*	*11.844*					

TABLE 8.8: Model results for the estimation of the BU6_LOS durations.

8.4 NM1_step

The next production phase is mixing, which results in the NM1_step durations. The results of the grid search on the estimation of the NM1_step durations are shown in Table 8.9.

Model	**Feature set**	**Hyperparameters**	**MAE**	**Std**
Linear Regression	5	fit_intercept = False	8.407	13.619
Robust Regression	3		8.000	16.651
Extra Trees	7	max_depth = 30 min_samples_leaf = 10	6.866	17.759
LightGBM	5	max_depth = 10 min_data_in_leaf = 100 num_leaves = 200 n_estimators = 500 learning_rate = 0.05 lambda_l2 = 0	6.637	14.799
Neural Network	1	n_layer1 = 8 n_layer2 = 0 dropout_input_layer = 0 dropout_hidden_layer = 0	8.035	16.857

TABLE 8.9: Grid search results for the estimation of the NM1_step durations.

The standard deviation of the MAE of the four days in the validation set is high for the models in the grid search (around 15 seconds), see Table 8.9. The reason is that all models perform worse on one of the four days (MAE of approximately 40 seconds instead of approximately 7 seconds). A reason could be that this day still contains outliers that were not filtered out by the filters described in Section 4.3.

According to Table 8.10, the MAE of the benchmark is approximately 17 seconds. This model is not able to give an estimation for every batch due to its dependence on the occurrences of the same article. Approximately 3.8% of the batches could not be estimated.

Model	MAE	MAPE	Std	Number of predictions	Number of unknown predictions
Benchmark 1	16.985	13.727	3.788	12,770	509 (3.833%)
Benchmark 2	17.666	14.308	3.940	12,762	517 (3.893%)

TABLE 8.10: Benchmark results for the estimation of the NM1_step durations.

The proposed models in this research can estimate all batches. The results of the final models with the hyperparameters selected based on the performance in the grid search are shown in Table 8.11.

	With benchmark estimation		Without benchmark estimation		All		
Model	**MAE**	**MAPE**	**MAE**	**MAPE**	**MAE**	**MAPE**	**Std**
Linear Regression	10.543	8.297	13.481	10.889	10.657	8.398	3.350
Robust Regression	9.608	7.240	12.719	9.803	9.729	7.340	3.457
Extra Trees	11.836	9.436	14.708	11.828	11.948	9.529	4.521
LightGBM	**8.823**	6.755	**11.321**	9.047	**8.921**	6.844	3.984
Neural Network	9.020	6.626	12.549	9.433	9.158	6.736	3.484
Benchmark 2	*17.666*	*14.308*					

TABLE 8.11: Model results for the estimation of the NM1_step durations.

All models can improve the results on the same batches as were estimated by the benchmark, see Table 8.11, which answers the second sub-question of this research. The MAE of the models on the batches without benchmark estimation are smaller than the MAE of the benchmark, too. The overall performance of the LightGBM model is the best with an MAE of 8.921 seconds, which is approximately half of the MAE of the benchmark. These scores are measured over 78 days in the test set, which results in a standard deviation of 3.984 seconds for the LightGBM model.

Figure 8.6 shows the estimated and observed values for the benchmark and LightGBM. The improvement on the performance by the LightGBM model can be observed: the blue dots are closer to the grey diagonal line. In addition, the LightGBM is able to estimate the other batches (red dots in Figure 8.6).

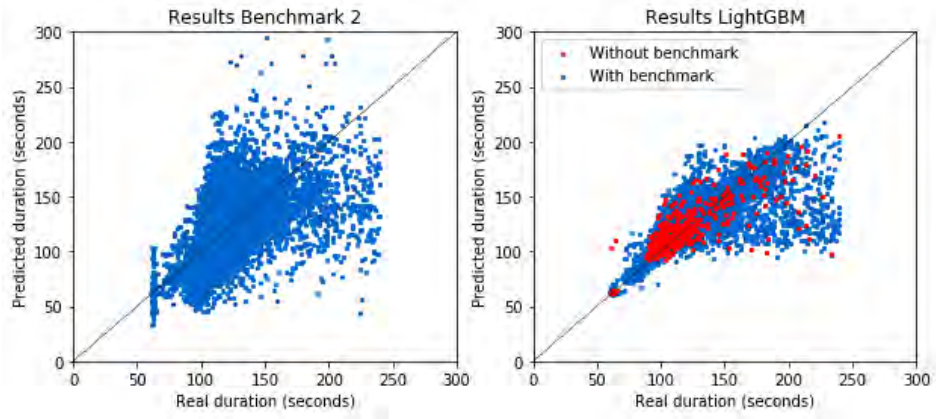


FIGURE 8.6: Estimated durations versus observed durations for the Benchmark (left) and proposed model (right) for the estimation of the NM1_step durations.

Note that both models are less accurate while the production duration increases, see Figure 8.6. On approximately 5% of the batches has the LightGBM model an absolute error

of more than 30 seconds. The benchmark has an absolute error of more than 30 seconds on approximately 18% of the batches.

The feature importance of the LightGBM model can be obtained, which is shown in Figure 8.7. According to Figure 8.7, the most important features are *dayNr*, *articleMax5*, *articleRecipeGroupMax5*, *articleGroupMax5* and *kgLiquids*. In Section 6 was shown how the NM1_step durations change over time, see Figure 6.10B. Therefore, the importance of the *dayNr* feature is not surprising. In contrast, the importance of the three article features is remarkable because the maximum duration is not robust to possible remaining outliers. The reason is likely to be related to the shape of Figure 8.6: large durations are more likely to be underestimated.

The most important ingredient for the LightGBM model is the ingredient *5721*, see Figure 8.7. However, all features of feature set 5 show some importance to the estimation of the mixing durations by the LightGBM model, which is the answer to the first research sub-question.

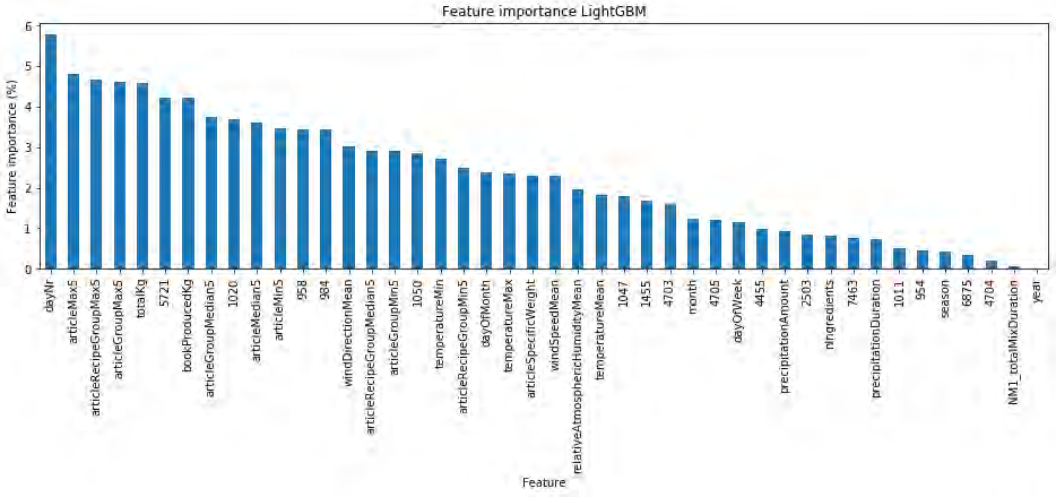


FIGURE 8.7: Feature importance LightGBM for the estimation of the NM1_step durations.

As mentioned before, it is relevant for ENGIE Industrial Automation how much training data is needed for the model to perform well. Therefore, the model performance is observed for less training data. The results are shown in Table 8.12.

	1 month			2 months		
Model	**MAE**	**MAPE**	**Std**	**MAE**	**MAPE**	**Std**
Linear Regression	12.093	9.726	8.572	10.889	8.604	4.163
Robust Regression	9.491	7.141	3.634	9.794	7.421	3.539
Extra Trees	10.146	7.944	4.555	10.355	8.120	4.545
LightGBM	**9.461**	7.307	3.558	**9.091**	6.970	3.882
Neural Network	10.219	7.609	4.113	10.120	7.617	3.849

	6 months			12 months		
Model	**MAE**	**MAPE**	**Std**	**MAE**	**MAPE**	**Std**
Linear Regression	10.620	8.304	3.216	10.527	8.249	3.304
Robust Regression	10.235	7.811	3.438	9.872	7.464	3.406
Extra Trees	10.759	8.496	4.495	11.017	8.756	4.514
LightGBM	**8.899**	6.816	3.699	**8.818**	6.746	3.746
Neural Network	9.979	7.500	3.552	9.354	6.859	3.638

TABLE 8.12: Model results for the estimation of the NM1_step durations by using smaller training sets.

According to Table 8.12, the LightGBM model is the best model for all tested amounts of data, and it is performing better when more data becomes available.

It is striking that the MAE of the Extra Trees model increases as more data becomes available. A reason could be that relationships in the features change over time and that therefore including more history to the model does not increase the performance. The importance of the feature *dayNr* in the LightGBM model suggests the same. However, according to the performances, the LightGBM can use this information better than the Extra Trees model.

Based on the results shown in Tables 8.10, 8.11 and 8.12, the LightGBM is the best model for the estimation of the NM1_step durations. It performs almost twice as good as the benchmark.

8.5 NM1_Los_step

The estimation of the NM1_Los_step durations is performed with a simple time-based model, as explained in Section 6. A grid search is performed to find the best number of batches in the time window. The best number of batches is 80, which results in an MAE of 0.690 seconds on the validation set. Based on this number of batches is the final model created.

Benchmarks 1 and 2 result in the same performance, because the median duration does not change for the samples in the test set (21 seconds). This results in an MAE of 5.417 seconds, see Table 8.13. The benchmark is able to estimate all batches.

Model	**MAE**	**MAPE**	**Std**	**Number of predictions**	**Number of unknown predictions**
Benchmark 1 and 2	5.417	20.407	0.186	14,397	0 (0%)

TABLE 8.13: Benchmark results for the estimation of the NM1_Los_step durations.

The model that is proposed in this research calculates the median duration of the last 80 batches. By using a moving window, changes in time are captured. Therefore, the proposed model is better in estimating the NM1_Los_step durations in the test set than the benchmark, see Figure 8.8.

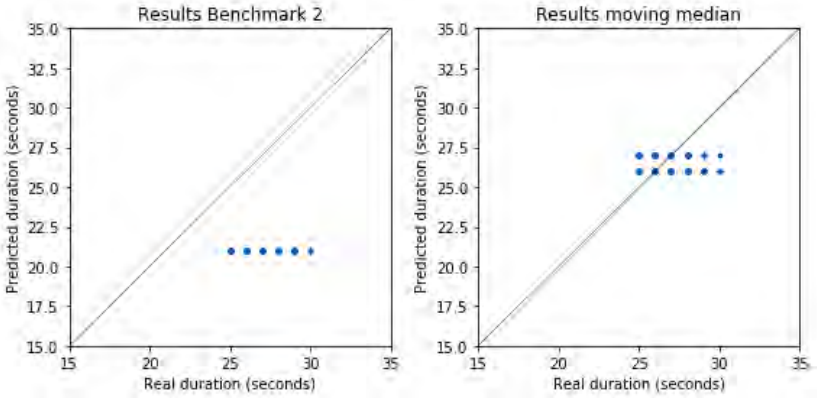


FIGURE 8.8: Estimated durations versus observed durations for the benchmark (left) and proposed model (right) for the estimation of the NM1_Los_step durations.

The benchmark model is not accurate for the test samples, because the distribution of the durations is changed over time, see Figure 4.16 in Section 4. Therefore, the blue dots are not close to the grey diagonal in the left plot of Figure 8.8.

The proposed model has an MAE of 0.728 seconds, see Table 8.14. Also, the standard deviation of the MAE over the 78 days in the test set is reduced to 0.092 seconds. Besides, the p-value of the t-test is much smaller than 0.01. Therefore, the proposed model performs significantly better in estimating the NM1_Los_step durations than the benchmark.

Model	**MAE**	**MAPE**	**Std**
Median duration of the last 80 batches	**0.728**	2.712	0.092
Benchmark 2	*5.417*	*20.407*	*0.186*

TABLE 8.14: Proposed model results for the estimation of the NM1_Los_step durations.

8.6 MML_AFV_TR

The MML_AFV_TR durations are the time it takes to empty the bunker below the mixer (BU7). A screw conveyor is used to transport the batch from BU7 to the transportation system. Therefore, the MML_AFV_TR durations are correlated to the batch size. The grid search is therefore performed for both the total duration, as for the duration per 1,000 kg (and multiplied with the batch size afterwards). This approach is similar to the grid search for the HA1_TOEV durations. However, the total durations happens now to be the best objective for all models instead of the duration per 1,000 kg. The results of the grid search are shown in Table 8.15.

Model	Feature set	Hyperparameters	MAE	Std
Linear Regression	4	fit_intercept = False	20.589	2.723
Robust Regression	4		20.140	2.994
Extra Trees	5	max_depth = 30 min_samples_leaf = 10	18.689	4.788
LightGBM	4	max_depth = 10 min_data_in_leaf = 10 num_leaves = 100 n_estimators = 300 learning_rate = 0.1 lambda_l2 = 0	17.950	5.643
Neural Network	1	n_layer1 = 8 n_layer2 = 8 dropout_input_layer = 0 dropout_hidden_layer = 0	21.281	3.113

TABLE 8.15: Grid search results for the estimation of the MML_AFV_TR durations.

According to Table 8.15, the LightGBM performs the best on the validation set. However, this model also has the biggest standard deviation over the four days. Based on the t-test, we can conclude that the performance of the LightGBM, Extra Trees and Robust Regression on the validation set is not significantly different (p-value t-test > 0.05). The best feature set and hyperparameters, shown if Table 8.15, are used for the final models.

Benchmarks 1 and 2 perform approximately the same on the test set, see Table 8.16. They have both a MAE of approximately 33 seconds (16.4%). Note that no benchmark estimation could be given for less than 0.5 percentage of the batches.

Model	MAE	MAPE	Std	Number of predictions	Number of unknown predictions
Benchmark 1	32.946	16.400	7.210	14,189	32 (0.225%)
Benchmark 2	32.962	16.412	7.190	14,161	60 (0.422%)

TABLE 8.16: Benchmark results for the estimation of the MML_AFV_TR durations.

In contrast, the final models can estimate all batches of the test set. The results of the final models on the test set are shown in Table 8.17. All evaluated models are able to outperform the benchmark estimations, see Table 8.17, which answers the second sub-question of this research.

	With benchmark estimation		Without benchmark estimation		All		
Model	**MAE**	**MAPE**	**MAE**	**MAPE**	**MAE**	**MAPE**	**Std**
Linear Regression	23.762	11.608	20.750	11.725	23.750	11.609	8.321
Robust Regression	23.670	11.549	21.179	12.068	23.659	11.551	8.365
Extra Trees	**20.861**	10.076	**17.888**	9.759	**20.848**	10.075	8.479
LightGBM	21.103	10.346	18.615	10.628	21.093	10.347	8.393
Neural Network	23.959	11.425	22.520	12.397	23.953	11.429	8.416
Benchmark 2	*32.962*	*16.412*					

TABLE 8.17: Model results for the estimation of the MML_AFV_TR durations.

According to Table 8.17, the Extra Trees model performs best with an MAE of approximately 20.8 seconds. In addition, the model performs best for batches with (20.9 seconds) and without benchmark estimation (17.9 seconds). According to Table 8.17, all models seem to perform better on the batches without benchmark estimation. However, these batches happen to have small durations and the models are performing worse when the duration increases, see the red dots in the right plot of Figure 8.9 for the batches without benchmark estimation.

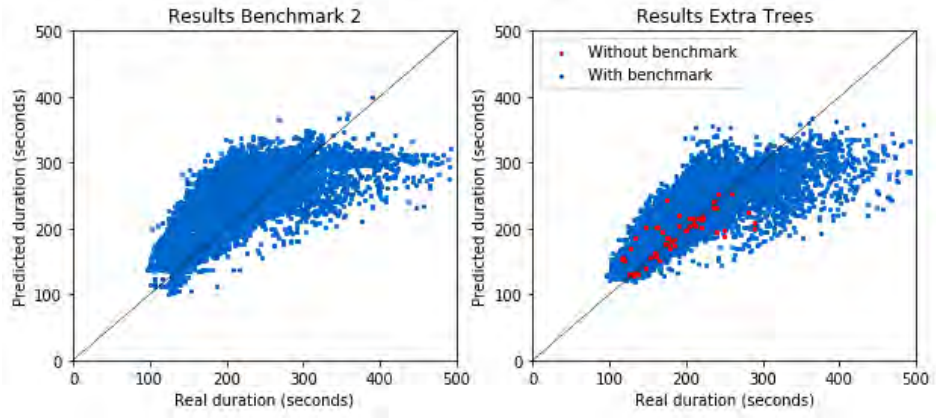


FIGURE 8.9: Estimated durations versus observed durations for the benchmark (left) and the Extra Trees model (right) for the estimation of the MML_AFV_TR durations.

Figure 8.9 shows that both the benchmark model and the Extra Trees are not able to accurately estimate large MML_AFV_TR durations. It might be the case that these durations are still outliers that were incorrectly kept in the data in the data filtering step of the research.

Similar to the other production durations, the amount of training data is evaluated. According to Table 8.18, the Extra Trees model is the best model for all evaluated training amounts. Therefore, the Extra Trees model is the best model for the estimation of the MML_AFV_TR durations.

	1 month			**2 months**		
Model	**MAE**	**MAPE**	**Std**	**MAE**	**MAPE**	**Std**
Linear Regression	23.307	11.327	8.437	23.293	11.288	8.546
Robust Regression	22.481	10.880	9.031	22.658	10.985	8.957
Extra Trees	**21.736**	10.331	8.413	**21.293**	10.149	8.569
LightGBM	23.204	11.169	8.606	22.624	10.936	8.330
Neural Network	24.405	11.549	9.036	24.213	11.606	8.987

	6 months			**12 months**		
Model	**MAE**	**MAPE**	**Std**	**MAE**	**MAPE**	**Std**
Linear Regression	23.239	11.264	8.671	23.297	11.305	8.414
Robust Regression	22.860	11.097	8.670	23.097	11.206	8.475
Extra Trees	**20.997**	10.063	8.570	**20.798**	10.015	8.493
LightGBM	21.765	10.576	8.336	21.390	10.430	8.419
Neural Network	24.571	11.971	8.399	24.049	11.530	8.454

TABLE 8.18: Model results with smaller training sets for the estimation of the MML_AFV_TR durations.

To see which features affect the MML_AFV_TR durations, the first sub-question of this research, the feature importances in the Extra Trees model are evaluated. The importances can be obtained by looking at which features are used for the splits in the Decision Trees of the Extra Trees model. The results are shown in Figure 8.10.

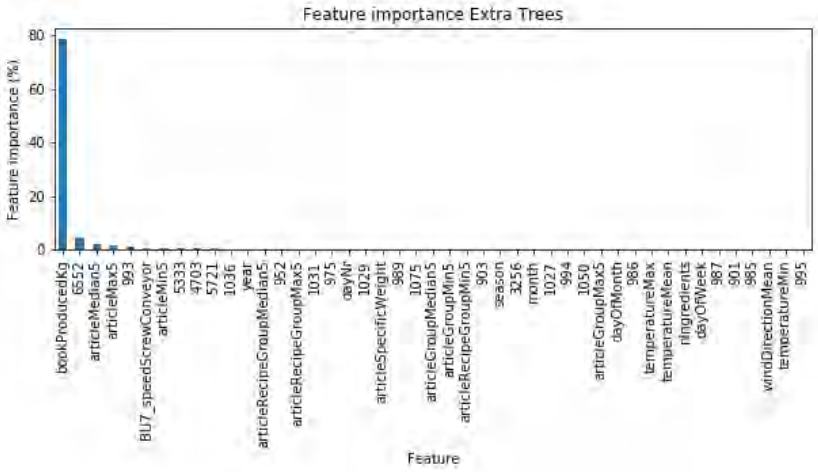


FIGURE 8.10: Feature importance Extra Trees for the estimation of the MML_AFV_TR durations. (200 features are not shown, because these are not used by the model.)

Figure 8.10 shows that the batch size (*bookProducedKg*) is very important in the estimation of the MML_AFV_TR durations. This is also what we expected according to the feature analysis in Section 6.

The ingredients *6552*, *993* and *5333* are the most important ingredients, according to Figure 8.10. These ingredients are also the ingredients with the largest positive Pearson correlation, see Figure A.4 in the Appendix.

As mentioned in Section 6, the speed of the screw conveyor is not frequently changed. Therefore, it is not surprising that this feature is not very important, see Figure 8.10.

8.7 MML_AFV_TR_NADRAAI

The seventh target variable, MML_AFV_TR_NADRAAI, is the transportation duration of GML batches to the next destination. The benchmark calculates the median of the duration of the last 31 batches that are transported to the same destination. The proposed model in this research is very similar, but now the number of batches is determined by performing a grid search. The best number of batches is found to be 600, which results in an MAE of 1.220 seconds on the validation set. Note that this 600 is the maximum window size, which means that fewer batches are used when less data is available.

Benchmarks 1 and 2 are able to estimate all 7,701 batches, see Table 8.19. They both have an MAE of approximately 1.24 seconds. The proposed model with a window size of 600 batches per destination has an MAE of 1.21 seconds, see Table 8.20.

Model	MAE	MAPE	Std	Number of predictions	Number of unknown predictions
Benchmark 1	1.238	1.165	0.132	7,701	0 (0%)
Benchmark 2	1.236	1.162	0.128	7,701	0 (0%)

TABLE 8.19: Benchmark results for the estimation of the MML_AFV_TR_NADRAAI durations.

Model	MAE	MAPE	Std
Median duration of the last 600 batches with the same destination	**1.214**	1.144	0.128
Benchmark 2	*1.236*	*1.162*	*0.128*

TABLE 8.20: Proposed model results for the estimation of the MML_AFV_TR_NADRAAI durations.

To test if this proposed model is significantly better than the benchmark model, the t-test is performed. The null hypothesis of the t-test is that the means of the absolute errors are the same. The p-value of the performed t-test is 0.22, which is bigger than any common significance level (0.05, 0.025 or 0.01). Therefore, the hypothesis is not rejected, which means that the performance of the proposed model and the benchmark are not significantly different. Both models could, therefore, be used for the estimation of the MML_AFV_TR_NADRAAI durations.

8.8 PL_PO1

The pressing duration of the first press line, PL_PO1, is the next target variable. As described in Section 4.3.2 and Section 7, the pressing duration is estimated by predicting PL_PO1_intercept and PL_PO1_slope separately, see Table 4.3. The results are obtained by comparing the estimated total pressing duration with the observed total pressing duration.

The results of the grid search are shown in Table 8.21. According to Table 8.21, all best grid points use a different feature set.

Model	Feature set	Hyperparameters	MAE	Std
Linear Regression	2	fit_intercept = True	169.968	60.180
Robust Regression	4		167.710	69.522
Extra Trees	5	max_depth = 20 min_samples_leaf = 10	148.473	73.705
LightGBM	6	max_depth = 8 min_data_in_leaf = 10 num_leaves = 50 n_estimators = 100 learning_rate = 0.05 lambda_l2 = 0	142.777	61.844
Neural Network	1	n_layer1 = 8 n_layer2 = 0 dropout_input_layer = 0 dropout_hidden_layer = 0	167.895	61.183

TABLE 8.21: Grid search results for the estimation of the PL_PO1 durations.

The final models will be compared to the benchmark on the test set. Benchmarks 1 and 2 have approximately the same MAE on the test set: 240 seconds (9.3%). Note that the GML batches for the same order are processed continuously on the press line. Therefore, there is less difference between Benchmark 1 and Benchmark 2 than at the GML. Both benchmarks could not estimate 216 batches (12.617%) because the Robust Regression model, which is used by the benchmark, needs at least 5 batches of the same article.

Model	**MAE**	**MAPE**	**Std**	**Number of predictions**	**Number of unknown predictions**
Benchmark 1	239.589	9.314	229.938	1,496	216 (12.617%)
Benchmark 2	239.606	9.315	229.938	1,496	216 (12.617%)

TABLE 8.22: Benchmark results for the estimation of the PL_PO1 durations.

The results of the final models are shown in Table 8.23. According to Table 8.23, all evaluated models are able to outperform the benchmark estimations, which is the answer to the second sub-question of this research. It shows that the Extra Trees model performs best, which results in an MAE of approximately 174.5 seconds (6.8%), see Table 8.23. Extra Trees performs best on both batches with and without benchmark estimations. Note that this model is able to estimate all batches in the test set.

	With benchmark estimation		**Without benchmark estimation**		**All**		
Model	**MAE**	**MAPE**	**MAE**	**MAPE**	**MAE**	**MAPE**	**Std**
Linear Regression	194.466	7.485	186.814	7.148	193.501	7.442	213.849
Robust Regression	182.760	7.200	188.329	7.362	183.462	7.220	212.544
Extra Trees	**173.427**	6.760	**181.607**	6.840	**174.459**	6.770	186.964
LightGBM	179.569	7.062	193.269	7.158	181.298	7.074	163.199
Neural Network	190.586	7.413	208.392	7.830	192.832	7.466	219.969
Benchmark 2	*239.606*	*9.315*					

TABLE 8.23: Model results for the estimation of the PL_PO1 durations.

The Neural Network only uses feature set 1 and is still able to show reasonably good results. One reason for this performance is that the PL_PO1 duration primarily depends on which press units are used, which is for approximately 87% of the batches only press unit B (Section 6). Since almost all batches use the same press unit, the Neural Network is able to show good results, despite the setpoints are not included.

The estimations of the benchmark and Extra Trees model are shown in Figure 8.11. The blue dots show the batches with benchmark estimation and the red dots the batches without benchmark estimation. The grey diagonal indicates a perfect fit between the estimations and observations. The dots created by Extra Trees are closer to the diagonal than the dots created by the benchmark, see Figure 8.11. Therefore, the improvement made by Extra Trees can also be observed in this way.

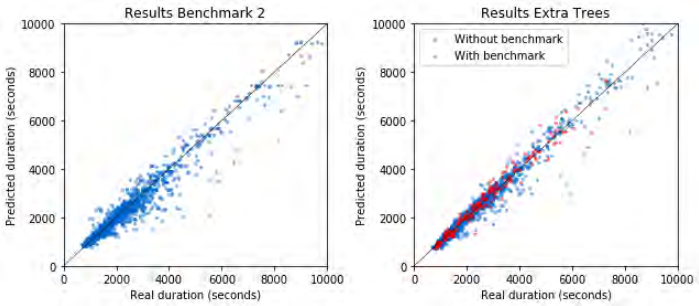


FIGURE 8.11: Estimated durations versus observed durations for the benchmark (left) and the Extra Trees model (right) for the estimation of the PL_PO1 durations.

The required amount of training data can be evaluated based on the results in Table 8.24. The Extra Trees model happens to be still the best model when less training data is available, see Table 8.24. Therefore, the Extra Trees model is the best model for the estimation of the PL_PO1 durations.

	1 month			**2 months**		
Model	**MAE**	**MAPE**	**Std**	**MAE**	**MAPE**	**Std**
Linear Regression	189.161	7.385	222.161	188.897	7.351	209.056
Robust Regression	215.431	8.614	239.610	198.869	7.731	364.882
Extra Trees	**186.400**	7.214	193.578	**181.353**	7.012	188.030
LightGBM	208.191	7.942	186.517	205.281	7.845	174.728
Neural Network	214.071	8.226	178.519	201.418	7.812	218.866

	6 months			**12 months**		
Model	**MAE**	**MAPE**	**Std**	**MAE**	**MAPE**	**Std**
Linear Regression	188.445	7.203	203.332	195.651	7.404	202.326
Robust Regression	185.068	7.239	197.602	185.332	7.219	214.374
Extra Trees	**179.786**	6.949	189.168	**175.608**	6.800	196.240
LightGBM	194.331	7.509	174.060	184.736	7.169	166.906
Neural Network	195.528	7.487	210.627	195.005	7.476	208.592

TABLE 8.24: Model results with smaller training sets for the estimation of the PL_PO1 durations.

The features that are most important for the Extra Trees model can be observed in Figure 8.12. Only the features with an importance of more than 0.2% are shown in order to limit the number of bars in the figure.

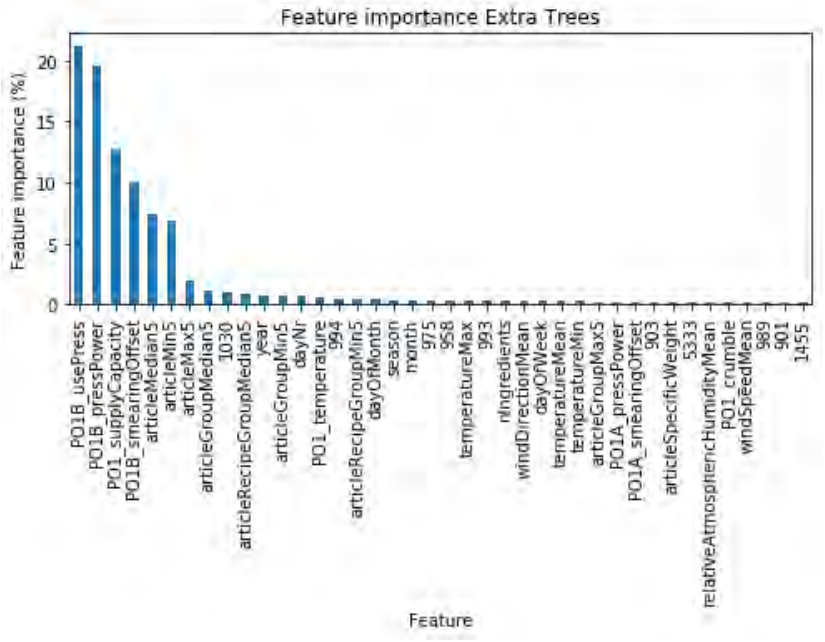


FIGURE 8.12: Feature importance Extra Trees for the estimation of the PL_PO1 durations. (210 features are not shown, because these are not used by the model.)

There are two features indicating if press unit A and/or press unit B is used. However, only the feature *PO1B_usePress* is very important to the Extra Trees model, see Figure 8.12. The reason for this is that before the first of January 2018, only press unit B or both press units were used, and after then, either press unit A or B. Therefore, using a time feature (for example *year*) and the usage of press unit B is enough to obtain the same information.

Since press unit B is the most used press unit (90%), the features belonging to this unit are also more important to the model, see Figure 8.12. The ingredients do not seem to be very important for the estimation of the pressing duration at PL 1. Figure 8.12 shows that the setpoints are most important features for the Extra Trees model, which is the answer to the first sub-question of this research.

8.9 PL_PO2

The pressing duration at PL 2, PL_PO2, is the ninth target variable of this research. The same approach is used as for the estimations of the PL_PO1 durations. A grid search is performed to find the best feature set and hyperparameters. The results are shown in Table 8.25.

Model	Feature set	Hyperparameters	MAE	Std
Linear Regression	6	fit_intercept = False	181.426	68.802
Robust Regression	6		181.190	69.713
Extra Trees	6	max_depth = 20 min_samples_leaf = 10	168.676	63.318
LightGBM	6	max_depth = 8 min_data_in_leaf = 10 num_leaves = 100 n_estimators = 100 learning_rate = 0.1 lambda_l2 = 1	155.415	47.808
Neural Network	2	n_layer1 = 8 n_layer2 = 8 dropout_input_layer = 0 dropout_hidden_layer = 0.2	173.935	68.386

TABLE 8.25: Grid search results for the estimation of the PL_PO2 durations.

Note that for each model, except for the Neural Network, feature set 6 is selected, see Table 8.25. The final models are created with the hyperparameters and feature set shown in Table 8.25. The benchmark model will be compared to the final models, to evaluate the performance.

Similar to the estimation of the PL_PO1 durations, Benchmarks 1 and 2 show very similar results, see Table 8.26. The MAE of the benchmark is approximately 221.6 seconds (7.6%), and is not able to give an estimation for 84 batches (5.5%).

Model	MAE	MAPE	Std	Number of predictions	Number of unknown predictions
Benchmark 1	221.561	7.628	73.684	1,446	84 (5.490%)
Benchmark 2	221.596	7.628	73.659	1,446	84 (5.490%)

TABLE 8.26: Benchmark results for the estimation of the PL_PO2 durations.

The results of the final models are shown in Table 8.27, which shows that all evaluated models are able to outperform the benchmark estimations, which answers the second subquestion of this research. Table 8.27 shows that the Robust Regression model is the best overall. However, the LightGBM performs best on the batches without benchmark estimation.

	With benchmark estimation		Without benchmark estimation		All		
Model	**MAE**	**MAPE**	**MAE**	**MAPE**	**MAE**	**MAPE**	**Std**
Linear Regression	158.352	5.985	244.346	9.389	163.073	6.172	68.558
Robust Regression	**151.361**	5.661	248.668	9.468	**156.703**	5.870	69.291
Extra Trees	154.296	5.909	230.513	8.843	158.481	6.070	56.267
LightGBM	163.825	6.063	**225.471**	8.463	167.209	6.195	74.214
Neural Network	163.744	6.116	253.639	9.672	168.679	6.311	66.630
Benchmark 2	*221.596*	*7.628*					

TABLE 8.27: Model results for the estimation of the PL_PO2 durations.

From Table 8.28 can be obtained that the Extra Trees model is the best model to estimate the PL_PO2 durations with less than 12 months of training data. The Robust Regression model outperforms the Extra Trees model when 12 months or more training data is available. Note that the standard deviation of the daily Extra Trees models (56.3 seconds) is smaller than the standard deviation of the daily Robust Regression models (69.3 seconds), see Table 8.27. Besides, the Extra Trees model is the second best model to estimate the PL_PO2 durations using the full training dataset, see Table 8.27.

	1 month			**2 months**		
Model	**MAE**	**MAPE**	**Std**	**MAE**	**MAPE**	**Std**
Linear Regression	270.229	9.304	363.179	183.340	6.544	121.287
Robust Regression	220.754	7.713	208.809	175.800	6.209	126.802
Extra Trees	**163.425**	6.096	57.528	**161.811**	6.069	54.775
LightGBM	194.244	6.950	93.979	185.750	6.699	75.450
Neural Network	196.586	6.919	86.982	192.368	6.804	77.060

	6 months			**12 months**		
Model	**MAE**	**MAPE**	**Std**	**MAE**	**MAPE**	**Std**
Linear Regression	171.692	6.425	64.128	160.278	6.059	62.896
Robust Regression	161.697	5.983	62.352	**155.507**	5.753	67.913
Extra Trees	**159.271**	6.053	54.405	156.121	5.984	56.292
LightGBM	177.793	6.445	88.642	170.016	6.300	69.785
Neural Network	184.255	6.601	83.532	173.618	6.295	73.887

TABLE 8.28: Model results with smaller training sets for the estimation of the PL_PO2 durations.

The percentage error made by the two models can be compared with the percentage error of the benchmark, see Figure 8.13. Note that the peak of Extra Trees is not located at zero, see Figure 8.13. This suggests that the predictions of the Extra Trees are influenced by the batches that took unexpectedly long: it overestimates a little bit (positive percentage error in Figure 8.13), to reduce the underestimation of a few batches (negative percentage error in Figure 8.13). This can also be observed to a lesser extent for the benchmark and Robust Regression, see Figure 8.13. They seem to be more robust to these outliers.

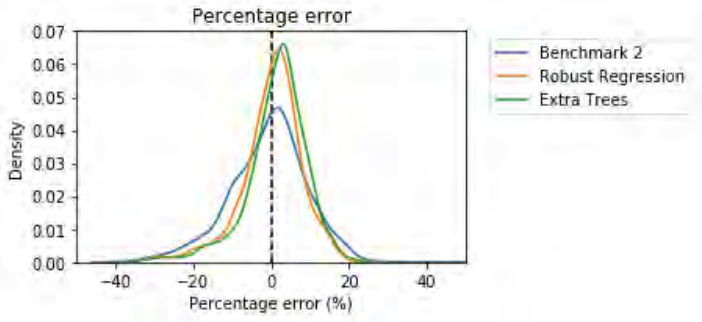


FIGURE 8.13: Density distribution of the percentage error of the benchmark, Extra Trees and Robust Regression for the estimation of the PL_PO2 durations.

Both the density plot of the percentage error made by the Robust Regression and the Extra Trees show less spread, i.e., smaller width, than the percentage error made by the benchmark, see Figure 8.13. The standard deviation of the percentage error made by the Robust Regression, Extra Trees and benchmark are respectively 7.944, 7.878 and 9.987 seconds.

Since both the Extra Trees and Robust Regression show good results (small MAE), the importance of the features will be evaluated for both models to answer the first sub-question of this research.

The coefficients trained by the Robust Regression are shown in Figure 8.14. From these coefficients can be observed that the different ingredient article groups have the largest negative coefficients. Since the sum of the ingredient article groups is 100% of the batch, we can observe that if the batch contains more liquids (*igrGroup_VL*), it results into a smaller pressing duration. Besides the ingredient article groups, the *supply capacity* shows the largest negative coefficient, see Figure 8.14.

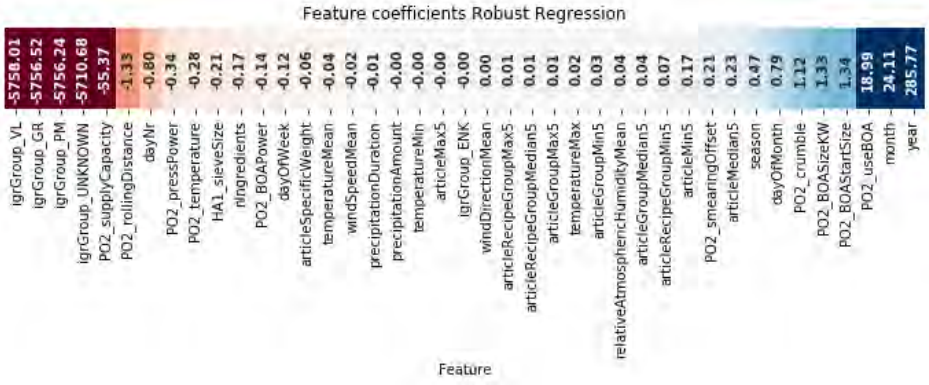


FIGURE 8.14: Feature importance Extra Trees for the estimation of the PL_PO2 durations.

In addition, increasing the rolling distance (*PO2_rollingDistance*) reduces the pressing duration and the usage of the BOA (*PO2_useBOA*) increases the PL_PO2 durations, according to Figure 8.14. However, according to the importance of the features in the Extra Trees, the usage of the BOA (*PO2_useBOA*) and the rolling distance (*PO2_rollingDistance*) have not much effect on the PL_PO2 durations, see Figure 8.15.

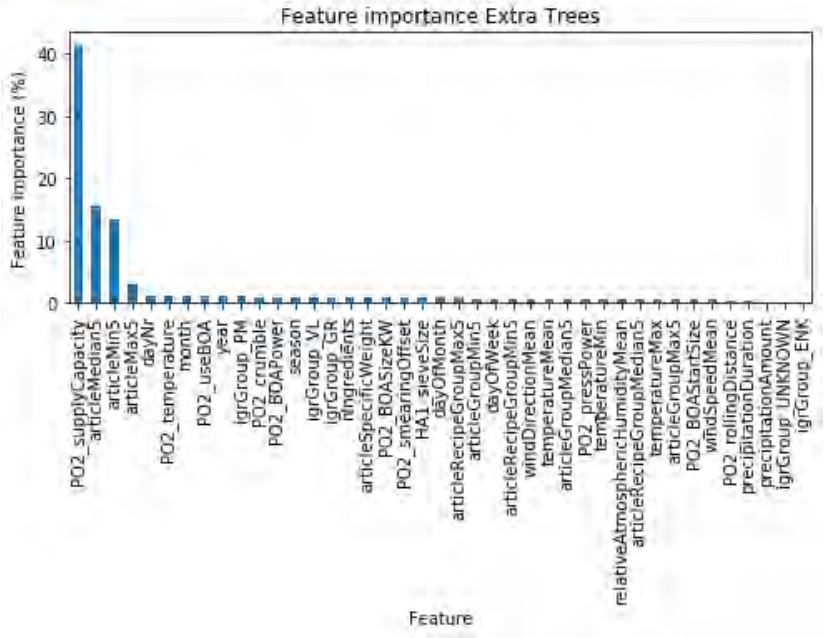


FIGURE 8.15: Feature importance Extra Trees for the estimation of the PL_PO2 durations.

Feature *igrGroup_VL* is the most important ingredient article feature according to Extra Trees, which was also observed from the Robust Regression coefficients. However, its importance is much smaller in the Extra Trees model than in the Robust Regresssion model. The most important features for the Extra Trees model are the supply capacity (*PO2_supplyCapacity*) and the article features, see Figure 8.15.

8.10 PL_PO3

The estimation of the PL_PO3 durations is comparable with the estimation of the other pressing durations. However, PL 3 has two parallel press units, which is different from the other press lines. Table 8.29 shows the best grid point found for each machine learning model during the grid search. Note that feature set 6 is selected by all models. This suggests that the ingredient article groups are more useful than the individual ingredient percentages. This will be examined during the evaluation of the feature importances.

Model	Feature set	Hyperparameters	MAE	Std
Linear Regression	6	fit_intercept = True	142.378	44.246
Robust Regression	6		132.395	46.783
Extra Trees	6	max_depth = 20 min_samples_leaf = 1	112.695	17.837
LightGBM	6	max_depth = 10 min_data_in_leaf = 50 num_leaves = 100 n_estimators = 300 learning_rate = 0.1 lambda_l2 = 0	95.749	11.683
Neural Network	6	n_layer1 = 4 n_layer2 = 0 dropout_input_layer = 0 dropout_hidden_layer = 0	134.952	50.383

TABLE 8.29: Grid search results for the estimation of the PL_PO3 durations.

The benchmark is evaluated similarly as before by looking at the results of Benchmark 1 (uses all batches to estimate the next batch) and Benchmark 2 (uses all batches of the days before the day of the next batch). Benchmarks 1 and 2 show again very similar results. They have an MAE of approximately 313 seconds (11%). Note that this benchmark performs worse compared to the benchmark for the other pressing durations. Also, the benchmark is not able to estimate 70 durations (5%) of the test set.

Model	MAE	MAPE	Std	Number of predictions	Number of unknown predictions
Benchmark 1	313.148	11.052	218.232	1,318	70 (5.043%)
Benchmark 2	313.241	11.055	218.257	1,318	70 (5.043%)

TABLE 8.30: Benchmark results for the estimation of the PL_PO3 durations.

The final models are evaluated on the test set, see Figure 8.31 for the results. Note that all MAEs on the test set (Figure 8.31) are significantly larger than the MAEs in the grid search (Figure 8.29). The reason for this is that the grid search consists of only four days, which

happen to be easier to estimate than the days in the test set. Since the test set consists of 78 days, the MAEs on the test set are more representative.

	With benchmark estimation		**Without benchmark estimation**		**All**		
Model	**MAE**	**MAPE**	**MAE**	**MAPE**	**MAE**	**MAPE**	**Std**
Linear Regression	190.137	5.523	128.835	5.560	187.046	5.525	112.385
Robust Regression	180.694	5.223	124.620	5.394	177.866	5.232	132.810
Extra Trees	**153.616**	4.600	**106.482**	4.571	**151.239**	4.598	142.579
LightGBM	161.134	4.743	119.954	4.887	159.057	4.750	177.839
Neural Network	327.255	10.067	463.241	12.012	334.113	10.165	306.936
Benchmark 2	*313.241*	*11.055*					

TABLE 8.31: Model results for the estimation of the PL_PO3 durations.

Besides, it is remarkable that the Neural Network is not able to outperform the benchmark, in contrast to the other machine learning models. Therefore, the answer to the second sub-question of this research is that all evaluated models, except for the Neural Network, are able to outperform the benchmark estimations. A reason for the performance of the Neural Network could be that the one layer of the Neural Network is not able to combine all features of feature set 6 in such a way that the MAE is reduced. Note that the difference between the Neural Network and the regression models is (apart from the 4 neurons instead of 1) the ReLU activation function. Therefore, it might be interesting to try other activation functions in future research.

Except for the Neural Network, all machine learning models can significantly improve the PL_PO3 duration estimations compared to the benchmark. One of the reasons for this improvement is that the machine learning models are able to use the information if the batch uses both parallel press units or just a single one. When only a single press unit is used, the production duration is approximate twice the pressing duration with both press units. The benchmark is not able to use this information, because it only takes the batch size and the produced article into account.

The models are evaluated against the same test set, but now the amount of training data is reduced to give an idea of how much training data is required for good PL_PO3 duration estimations. The results are shown in Table 8.32.

	1 month			2 months		
Model	**MAE**	**MAPE**	**Std**	**MAE**	**MAPE**	**Std**
Linear Regression	226.859	6.736	392.014	» 1,000	» 1,000	» 1,000
Robust Regression	202.539	6.228	463.859	161.143	4.865	261.735
Extra Trees	**159.656**	4.804	176.321	**160.010**	4.820	170.570
LightGBM	277.319	8.680	325.743	196.698	5.973	333.565
Neural Network	452.029	13.391	529.343	338.766	10.196	346.857

	6 months			12 months		
Model	**MAE**	**MAPE**	**Std**	**MAE**	**MAPE**	**Std**
Linear Regression	175.324	5.267	141.535	174.388	5.255	116.902
Robust Regression	164.002	4.902	159.263	166.921	5.015	125.573
Extra Trees	**156.076**	4.697	141.142	**153.542**	4.624	128.107
LightGBM	161.382	4.868	114.336	161.729	4.828	132.100
Neural Network	308.153	9.369	280.482	268.489	8.226	376.420

TABLE 8.32: Model results with smaller training sets for the estimation of the PL_PO3 durations.

According to Table 8.32, the Extra Trees model performs the best for all evaluated amounts of training data. Note that the Linear Regression model has a very large ($>> 1,000$) MAE for the usage of two months of training data. A reason for this is that if an ingredient is used in the test set that was not frequently used before (almost always zero) in the training set, the coefficient trained on the training set can be very inaccurate. Multiplying to a very inaccurate coefficient could result in such a wrong estimation. This will be further discussed in the discussion (Section 9).

Based on the above-mentioned results, the Extra Trees model is recommended for the PL_PO3 estimations. The feature importances of Extra Trees are shown in Figure 8.16.

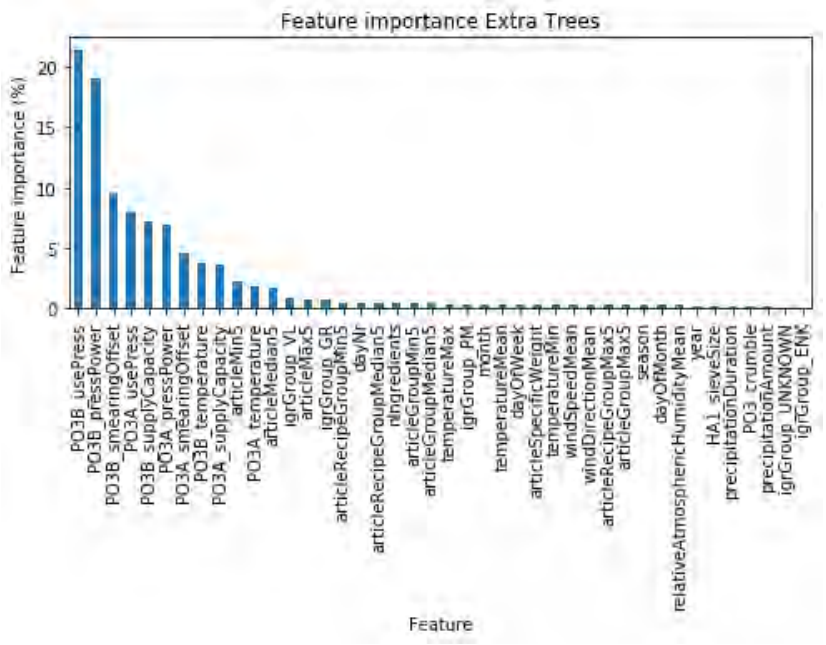


FIGURE 8.16: Feature importance Extra Trees for the estimation of the PL_PO3 durations.

According to Figure 8.16, the most important ingredient article group are the liquids (*igrGroup_VL*). Furthermore, all setpoints of the press units are important according to Figure 8.16. Only the setpoint *PO3_crumble* is not found to be important, which only indicates if the batch will be crumbled afterwards or not. The Extra Trees model is able to find the same features to be important as expected by ENGIE Industrial Automation, which is the answer to the first sub-question of this research. Therefore, we can conclude that the Extra Trees model is able to combine the features in such a way that it accurately estimates the PL_PO3 durations.

8.11 KO3_idle_time

The eleventh target variable is the cooling duration at press line 3: the KO3_idle_time duration. As explained in Section 6, the proposed model will be a simple time-based model, which estimates the production duration by calculating the median duration of the last B batches by filtering if the batch uses the crumbler or not. The best number of batches in the time window is found to be 100, which results in an MAE of 7.080 seconds on the validation set. Therefore, the proposed model will estimate the KO3_idle_time durations by calculating the median duration of the last 100 batches which also use or do not use the crumbler.

To test if the proposed model is better than the benchmark, both models will estimate the duration in the test set. Benchmarks 1 and 2 show very similar results, see Table 8.33. They have an MAE of approximately 8 seconds (7%). In addition, they are both able to estimate all cooling durations of the test set, see Table 8.33.

Model	**MAE**	**MAPE**	**Std**	**Number of predictions**	**Number of unknown predictions**
Benchmark 1	7.729	7.057	2.713	1,327	0 (0%)
Benchmark 2	7.843	7.163	2.739	1,327	0 (0%)

TABLE 8.33: Benchmark results for the estimation of the KO3_idle_time durations.

The proposed model reduces the MAE of the benchmark with approximately 1 second, see Table 8.34 compared to Table 8.33. The difference in the performance of the benchmark and the proposed model can be observed from Figure 8.17: the proposed model fits the grey diagonal a bit better.

Model	**MAE**	**MAPE**	**Std**
Median duration of the last 100 batches per usage of crumbler	**6.891**	6.115	2.659
Benchmark 2	*7.843*	*7.163*	*2.739*

TABLE 8.34: Proposed model results for the estimation of the KO3_idle_time durations.

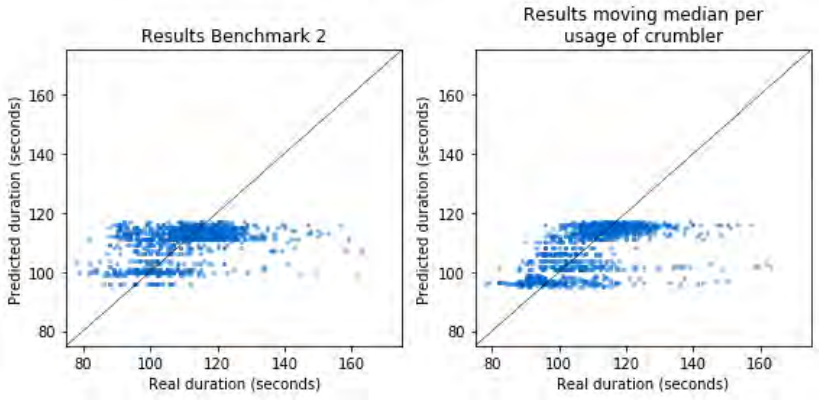


FIGURE 8.17: Estimated durations versus observed durations for the Benchmark (left) and proposed model (right) for the estimation of the KO3_idle_time durations.

The t-test is used to test if the difference between the MAE of the proposed model and the benchmark is significant. The null hypothesis is that the mean of the absolute errors of both models is equal. The p-value of the t-test is found to be approximately 0.0009, which is smaller than 0.01. Therefore, the null hypothesis is rejected, which means that the MAE of the proposed model and the benchmark are significantly different. The proposed model is thus significantly better than the benchmark.

8.12 PL_PO4

The PL_PO4 durations are the last values to be estimated in this research. The same approach is used as for the other press lines. This press line is most similar to PL 1. However, according to the feature analysis of Section 6, PL_PO1 mostly depends on the maximum power of the press, and PL_PO4 on the capacity of the supply.

The results of the grid search are shown in Table 8.35. The tree-based models show the smallest standard deviation on the four days in the validation set, see Table 8.35.

Model	**Feature set**	**Hyperparameters**	**MAE**	**Std**
Linear Regression	4	fit_intercept = False	128.165	25.522
Robust Regression	4		121.349	24.355
Extra Trees	6	max_depth = 30 min_samples_leaf = 10	122.634	8.498
LightGBM	6	max_depth = 8 min_data_in_leaf = 10 num_leaves = 50 n_estimators = 100 learning_rate = 0.1 lambda_l2 = 0	111.928	7.390
Neural Network	2	n_layer1 = 4 n_layer2 = 0 dropout_input_layer = 0 dropout_hidden_layer = 0	120.796	22.103

TABLE 8.35: Grid search results for the estimation of the PL_PO4 durations.

Benchmarks 1 and 2 have approximately the same MAE on the test set of approximately 213 seconds (8%), see Table 8.36. They are not able to give an estimation for 5.6% of the batches.

Model	**MAE**	**MAPE**	**Std**	**Number of predictions**	**Number of unknown predictions**
Benchmark 1	212.687	8.012	92.538	1,761	104 (5.576%)
Benchmark 2	212.792	8.016	92.529	1,761	104 (5.576%)

TABLE 8.36: Benchmark results for the estimation of the PL_PO4 durations.

All proposed machine learning models can give an estimation for all batches in the test set. The selected feature set and hyperparameters are shown in Table 8.35. The results of the final models on the test set are shown in Table 8.37.

	With benchmark estimation		**Without benchmark estimation**		**All**		
Model	**MAE**	**MAPE**	**MAE**	**MAPE**	**MAE**	**MAPE**	**Std**
Linear Regression	**142.719**	5.523	**144.467**	5.403	**142.816**	5.516	79.471
Robust Regression	144.354	5.546	148.050	5.463	144.560	5.541	80.745
Extra Trees	146.227	5.674	163.579	6.049	147.195	5.695	79.828
LightGBM	145.094	5.578	150.603	5.527	145.401	5.575	88.219
Neural Network	156.917	6.159	174.241	6.407	157.883	6.173	95.305
Benchmark 2	*212.792*	*8.016*					

TABLE 8.37: Model results for the estimation of the PL_PO4 durations.

Table 8.37 shows that all evaluated models are able to outperform the benchmark estimations, which is the answer to the second sub-question of this research. According to the results shown in Table 8.37, a linear model is most appropriate to estimate the PL_PO4_slope values of the pressing durations. The results suggest that the Linear Regression model performs better than the other models. However, when we test if the differences are significant, only the performance of the Neural Network is significantly different from the others using the t-test with a significance level of 0.05. Note that the Linear Regression model is very sensitive to features that are almost always zero, as explained in Section 8.10.

For the implementation, it is important how much training data is required for the models to perform well. The results of the final models using less training data are shown in Table 8.38.

	1 month			2 months		
Model	**MAE**	**MAPE**	**Std**	**MAE**	**MAPE**	**Std**
Linear Regression	» 1,000	» 1,000	» 1,000	» 1,000	» 1,000	» 1,000
Robust Regression	233.655	10.988	371.493	**151.597**	5.957	194.519
Extra Trees	**152.439**	5.790	80.746	153.291	5.842	81.540
LightGBM	164.862	6.276	92.461	165.116	6.295	94.522
Neural Network	177.036	6.664	98.948	177.638	6.662	542.320

	6 months			12 months		
Model	**MAE**	**MAPE**	**Std**	**MAE**	**MAPE**	**Std**
Linear Regression	» 1,000	» 1,000	» 1,000	143.611	5.569	77.836
Robust Regression	**136.567**	5.385	120.044	**141.817**	5.504	79.347
Extra Trees	152.010	5.797	83.789	149.069	5.734	78.996
LightGBM	153.039	5.858	83.017	150.261	5.781	93.153
Neural Network	157.318	5.980	129.984	156.894	5.964	91.458

TABLE 8.38: Model results with smaller training sets for the estimation of the PL_PO4 durations.

According to Table 8.38, the Linear Regression has a large MAE when the amount of data is reduced. This can be the result of the coefficient of ingredient features, which are almost always zero, see the explanation given for Table 8.32 in Section 8.10.

The Extra Trees model performs best on only one month of training data and has a smaller standard deviation than the other models. Robust Regression performs better when more data becomes available. However, as already mentioned, this difference was not found to be significant.

The coefficients of the Robust Regression model are shown in Figure A.7 in the Appendix. Since 203 features are included in feature set 4, the coefficients of the Robust Regression model are hard to interpret.

The importance of the features of the Extra Trees model are shown in Figure 8.18. Figure 8.18 shows that the article (*articleMedian5* and *articleMin5*) provides most information to the PL_PO4_slope estimations and after that the supply capacity (*PO4_supplyCapacity*), the press power of press unit A (*PO4A_pressPower*) and if press unit A is used (*PO4A_usePress*). Note that press unit B is always used at PL 4.

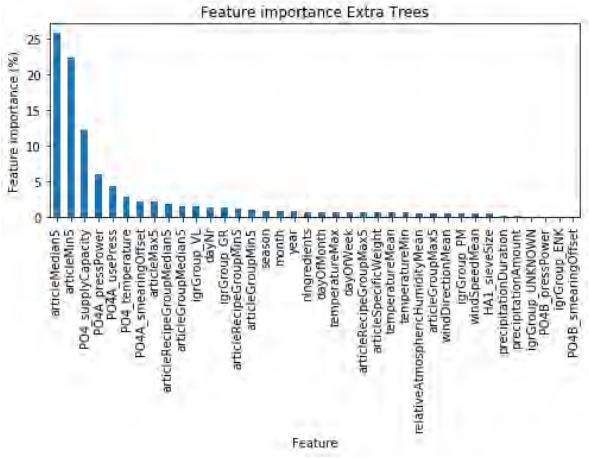


FIGURE 8.18: Feature importance Extra Trees for the estimation of the PL_PO4 durations.

Hence, similar features are important to the estimation of the PL_PO4 durations, as for the other pressing durations: the article features, the setpoints and the liquid percentage of the batch (*igrGroup_VL*), see Figure 8.18. This answers the first sub-question of this research.

8.13 Overview

Table 8.39 gives an overview of the above-mentioned results. The benchmark results in this overview are the Benchmark 2 results. These results are very similar to the Benchmark 1 results, except for HA1_TOEV. The best model results are shown in Table 8.39, which are the results of the best model on the full test set by training of the full training set.

Target variable	**Benchmark results**	**Best model results**	**Best model based on full training set**	**Best model based on small training set**
HA1_TOEV	20.289	16.293	NN, RR, LGBM	RR, ET, LGBM, NN
HA1_ZEEF	1.642	0.864	MD 70 per sieve switch	MD 70 per sieve switch
BU6_LOS	5.020	3.923	MD 50	MD 50
NM1_step	17.666	8.921	LGBM, NN	LGBM, RR
NM1_Los_step	5.417	0.728	MD 80	MD 80
MML_AFV_TR	32.962	20.848	ET, LGBM	ET, LGBM
MML_AFV_TR _NADRAAI	1.236	1.214	MD 600 per destination	MD 600 per destination
PL_PO1	239.606	174.459	ET, LGBM, RR	ET
PL_PO2	221.596	156.703	RR, ET, LGBM, NN	ET, RR
PL_PO3	313.241	151.239	ET, LGBM	ET, RR, LGBM
KO3_idle_time	7.843	6.891	MD 100 per usage of crumbler	MD 100 per usage of crumbler
PL_PO4	212.792	142.816	LR, RR, LGBM, ET	RR, ET

TABLE 8.39: Overview of final results. (LR=Linear Regression, RR=Robust Regression, ET=Extra Trees, LGBM=LightGBM, NN=Neural Network, MD=Median duration of last *B* batches)

The best models are listed in Table 8.39. The models listed for the same target variable are not significantly different based on the t-test with a significance level of 0.05. They are sorted based on the MAE score observed on the test set. Hence, it is significantly indifferent which of these models is used. According to Table 8.39, LightGBM is listed for every target variable by using the full training set. However, based on the smaller training sets, LightGBM is not always the best choice. Similar to LightGBM, the Neural Network is generally not the best choice when less training data is available. However, it performs best for the HA1_TOEV estimations.

From Table 8.39 can be observed that the Extra Trees and Robust Regression model show the best results on the smaller training sets. Since ENGIE Industrial Automation would like to use a similar approach for other industrial plants, it is recommended to implement these two models. It is recommended to use a Robust Regression model for the estimation of the

HA1_TOEV, NM1_step and PL_PO4 durations. The Extra Trees model would be the best model for the estimation of the MML_AFV_TR, PL_PO1, PL_PO2 and PL_PO3 durations.

Note that the proposed models all outperform the corresponding benchmark. Only the proposed model for the MML_AFV_TR_NADRAAI durations is not significantly better. Therefore, the answer to the research question is that it is possible to enhance production duration estimations by incorporating the ingredient composition of products, seasonal effects, and machine settings, for all production steps in plant X, except for the MML_AFV_TR-_NADRAAI durations.

9 Discussion

The goal of this research was to enhance the production duration estimations of an animal feed plant, which serve as input for a scheduling algorithm of ENGIE Industrial Automation. The research approach that was taken consisted of the following steps: problem understanding, data collection, feature creation, feature analysis, model creation and performance evaluation.

The first step, problem understanding, was very important in this research. From this step, the complexity of the problem became clear: what kind of variables do we have in an animal feed plant, how are they related, how is a batch processed and how could the data be retrieved. In addition, relevant literature was discussed to understand what kind of approaches were already taken and how this research could be innovative in that regard.

The second step in the research approach is the data collection step. The data was obtained from the database, which is also used by the software application which runs the industrial plant. Besides the data of plant X, a dataset from the KNMI website was downloaded to obtain weather data from the nearest weather station to plant X. According to the results of this research, the weather features have little impact on the production durations. Reasons could be that the weather data of the KNMI is not local enough, give only a summary of the whole day, and show the weather outside. Therefore, it would be interesting to use the temperature and humidity of inside the plant as model features, instead of the KNMI weather data.

After the dataset was obtained, it was cleaned. First of all, some batches were removed which were not reliable. Secondly, the outliers were removed. The outlier detection was performed for the production durations and the difference between the produced and requested batch size. However, no outlier detection was applied to other variables. It might be a good idea to do so in the future.

For outlier detection, the windowed IQR-rule was applied. The advantage of this outlier detection method is that it is easy to understand, and is easy to implement for future batches because it automatically detects shifts in the distribution. However, the disadvantage is that it takes some batches before it detects a shift in the distribution, which is a logical result of such a method. In addition, the best value for the window size might change in the future, and the selection of the least number of detected outliers does not guarantee the best value for the window size. Therefore, which and how many batches are filtered, should be tracked, so changes could be detected. One way of doing this would be to calculate the percentage of detected outliers in the past x days, and when this is more than some constant, a system expert is notified automatically.

The windowed IQR-rule was applied with $k = 1.5$ as lower bound and $k = 3$ as upper bound for the outlier detection of the production durations. This was selected based on the rule of thumb that value $k = 1.5$ filters suspected outliers and $k = 3$ filters real outliers. The distributions of the production durations are all very right-skewed (many batches took longer, but few shorter), therefore the selection of $k = 1.5$ as lower bound and $k = 3$ as upper bound was made. However, this is still a rule of thumb and it does not guarantee that only outliers are filtered and only non-outliers are kept. Future research could be done to find the best method for outlier detection.

The second step of the research approach also described the approach that was taken to separate the 'warm-up' period and period of full power or supply for the pressing durations. The assumption was made that the duration could be split by using the adjustment characteristic. This assumption seems fair because it uses domain knowledge and it reduces the complexity of the problem significantly. However, the 'warm-up' period is estimated by a Robust Regression model which is used in a windowed fashion. This approach introduces noise, because the Robust Regression model needs enough data points for an accurate estimation of the intercept (which is used as 'warm-up' period), and needs to be small enough to be able to track changes of the adjustment characteristic over time.

It would be interesting to investigate if the 'warm-up' period could be calculated accurately using the underlying parameters of the adjustment characteristic. This would eliminate the need for the Robust Regression model and its corresponding created noise. However, at this moment, this is not possible, because the underlying parameters are not saved for longer than 3 months. Therefore, for most batches in the data, these parameters are not known. From this point of view, it is strongly recommended to save this kind of data for a longer period.

In the third step of the research approach, features were created. The setpoint features were almost all similar to the database columns. Only the feature for the sieve switch was manually created. In addition, some article features were created, for which a window of 5 days was used. Other windows were not evaluated, because the total number of features was already very high. However, it would still be interesting for future research to evaluate the effect of other window sizes and aggregation functions (we just used minimum, maximum and median duration).

Two approaches were taken for the inclusion of the ingredient features. The first was to include all ingredient features separately by their percentage of the batch. In the second approach, the ingredient features were summed together per ingredient article group. However, other kinds of approaches could be taken too. For example, it would be interesting to use Principal Component Analysis (PCA) to reduce the number of ingredient features. This technique uses an orthogonal transformation to transform the possibly correlated ingredient features to linear uncorrelated features. This would be an interesting topic for future research.

When the non-batch related machine settings are saved for longer than 3 months, they could be used as features too. It would be interesting to see if the model performance increases when these settings are included. Therefore, ENGIE Industrial Automation is strongly recommended to save this kind of settings for longer than 3 months in the database.

The fourth step, feature analysis, showed the linear correlation between the different features and target features. The Pearson correlation was included in this report to measure the linear correlation. The Spearman rank correlation coefficient was calculated too. This correlation metric measures the monotonic relationship between variables, whether linear or not. However, both correlation measures showed approximately the same results. To limit the size of this report, it was decided to only include the Pearson correlation.

The features were analysed per feature group, because including all features would result in a very large matrix, which would be to complex to analyse. However, the correlation between feature groups would be interesting to examine too. Besides, the ingredient features were only compared to the production duration and not to each other with the same reason.

In the fifth step, model creation, the machine learning models were explained and a grid search was performed to find the best feature set and hyperparameters. Each grid point was

evaluated for four days in the validation set. However, the results suggest (high standard deviation and large differences with performance on the test set) that four days are too few to accurately perform the grid search. Therefore, when other values are tried, it would be better to use the full validation set for comparison. The argument that was given for the selection of four days, is that it takes a lot of computation time to run all the models for all different hyperparameters and feature sets, and for all 12 target durations of the animal feed plant. Note that for LightGBM more than 4,000 grid points were evaluated per target variable.

In the grid search, only seven feature sets were evaluated. This approach gives an idea of what kind of features are important. In addition, it is straight forward how this approach could be used in other animal feed plants, and how new ingredients and changes in the pilot plant (plant X) could be captured. However, the number of features can become very large when the number of ingredients keeps growing. In addition, many features are not important for the models, as can be observed from the feature importance figures in the results section (Section 8), and therefore result in noise for the models.

It would be interesting for future research to investigate the possibility of dynamic feature selection. For example, to dynamically select the top k most important features. This could be done in many ways. For example, by automatically selecting top k features based on the feature importance of a machine learning model or based on the strongest correlations of a Pearson or Spearman correlation matrix. The advantage of such an approach is that it will reduce the number of useless features and it keeps the number of features the same over time. However, the disadvantage of this approach is that the selected features can change over time, which could result in different behavior of the machine learning models.

In addition, it would be interesting to investigate other approaches to limit the number of ingredient features. As already mentioned, one common approach for feature reduction is Principal Component Analysis (PCA). It would be interesting to see what the effect will be on the performance of the models. The advantage of this approach is that the number of ingredient features is reduced by limiting the information that is thrown away. The disadvantage of this approach is that the created 'summarized' features are hard to interpret.

When the current approach of feature sets is implemented in practice, it would be good to select a number of required observations before a new ingredient is added as a feature. This is particularly important for Linear Regression, Robust Regression and the Neural Network because these models train weights for the features, which are likely to be wrong when only a few non-zero observations are present for a certain feature, which is the case for new ingredient features. The Extra Trees and LightGBM models are more robust to these features by their hyperparameter for the minimal number of samples in the leaves of their Decision Trees.

Finally, the best feature set and hyperparameters are selected for each model. These final models are evaluated on a test set. The results are compared to the benchmark. In addition, the model performance is evaluated for less amount of training data.

Robust Regression and Extra Trees have the best overall performance. However, the LightGBM performs best for the mixing durations (NM1_step) and the Neural Network for the grinding durations (HA1_TOEV). Remarkable is that the Neural Network performs worst on all other production durations than the grinding durations.

The early stopping technique that was used in this research, stopped the Neural Network from training when the MAE on the validation set did not decrease. However, it would be interesting to see if the performance of the Neural Network increases when the hyperparameters of the early stopping technique would be tuned. For example, we could set the hyperparameter 'patience', which is the number of epochs without improvement before we stop the training of the Neural Network, to more than one. In addition, we could tune

other hyperparameters of early stopping as well, like the minimum improvement. Finally, it would be interesting to try more complex Neural Networks and other types of activation functions, to see if the performance increases if it is able to create more complex relations.

In addition, the amount of training data was evaluated. For some models/production durations, it was concluded that the error of the model increased while adding more data points to the training set. This suggests that it would be better to 'forget' old observations. Therefore, it would be very relevant for ENGIE Industrial Automation to investigate how much training data is best for the models to both provide the ability to forget and the ability to learn. Another approach would be to examine the performance of a time-based model, for example, a Recurrent Neural Network. This model is designed to understand temporal behavior. Therefore, this would be an interesting topic for future research.

The Robust Regression and Extra Trees model show good results in this research. In general, the Extra Trees shows low standard deviation in the performance over the 78 days in the test set. In addition, the feature importances given by Extra Trees are clear. The model is able to use some relevant features and ignores others. In contrast, the LightGBM tries to use all features, which performs worse for most production durations using a smaller training set, which is probably due to over-fitting to many useless features. In general, for the production durations where the performance of the Extra Trees model is less accurate, the Robust Regression model performs well. Therefore, ENGIE Industrial Automation is advised to implement these two models and to use them for the production duration for which they are appropriate according to their performance described in Section 8.

The current models estimate the production durations for the 'normal' situation, for which the scheduling algorithm is used. However, it would be interesting to create a separate model for the prediction when something is likely to go wrong. This would be an interesting field of study for future research.

10 Conclusion

The research goal of this study is to enhance the production duration estimations by incorporating the ingredient composition of products, seasonal effects, and machine settings. For this purpose, machine learning models were created for each production step in the manufacturing process of plant X. This research shows that the proposed models estimate the production duration significantly better than the benchmark. Except for the duration estimation of the transportation system, which is as good as the benchmark.

In addition, two sub-questions are answered in this research. First of all, the explanatory variables of the production durations are both analysed during the feature analysis, as of which the importance is given in the proposed models. The second question is which models lead to better production duration estimations compared to the benchmark approach. From the results, we can conclude that all models estimate the durations for most production steps well. Only for the pressing duration estimation at press line 3, the Neural Network performed worse than the benchmark. In all other cases, all models outperformed the benchmark significantly.

In general, the Extra Trees model shows low standard deviation in the performance over the 78 days in the test set. In addition, the feature importances given by Extra Trees are clear. The model is able to use some relevant features and ignores others. In contrast, the LightGBM tries to use all features, which performs worse for most production durations using a smaller training set, which is probably due to the inclusion of many useless features. For the production durations where the performance of the Extra Trees model is less accurate, the Robust Regression model performs generally well. Therefore, ENGIE Industrial Automation is advised to implement these two models and to use them for the production duration for which they are appropriate according to their performance described in Section 8.

By improving the production duration estimation in this research, the scheduling algorithm of ENGIE Industrial Automation will be more accurate. Using the scheduling algorithm in plant X will probably reduce the total manufacturing time. The accuracy of the proposed schedules by the scheduling algorithm is very important, not only for its usefulness, but also for the trust of the employees of plant X in digital applications like this.

Bibliography

[1] D. Pollard, S. Chuo, and B. Lee, "Strategies for mass customization", *Journal of Business & Economics Research*, vol. 6, no. 7, pp. 77–86, 2008.

[2] T. M. Mitchell, *Machine Learning*. McGraw-Hill, 1997, p. 2, ISBN: 978-0070428072.

[3] *Different types of feed - mash, crumbles and pellets*, https://metzerfarms.blogspot.com/2017/11/different-types-of-feed-mash-crumbles.html, [Online; accessed 24-May-2019], 2017.

[4] A. ISA, *Isa-88.00. 01-2010 batch control part 1: Models and terminology*, 2010.

[5] J. Behnamian, "Survey on fuzzy shop scheduling", *Fuzzy Optimization and Decision Making*, vol. 15, no. 3, pp. 331–366, 2016.

[6] E. A. Toso, R. Morabito, and A. R. Clark, "Lot sizing and sequencing optimisation at an animal-feed plant", *Computers & Industrial Engineering*, vol. 57, no. 3, pp. 813–821, 2009.

[7] M. Pinedo, *Scheduling*. Springer, 2016, p. 434. DOI: 10.1007/978-3-319-26580-3.

[8] S. Goren and I. Sabuncuoglu, "Robustness and stability measures for scheduling: Single-machine environment", *IIE Transactions*, vol. 40, no. 1, pp. 66–83, 2008.

[9] B. Tadayon and J. C. Smith, "Algorithms and complexity analysis for robust single-machine scheduling problems", *Journal of Scheduling*, vol. 18, no. 6, p. 576, 2015.

[10] J.-S. Yao and F.-T. Lin, "Constructing a fuzzy flow-shop sequencing model based on statistical data", *International journal of approximate reasoning*, vol. 29, no. 3, pp. 215–234, 2002.

[11] X. Zhang, G. Yan, W. Huang, and G. Tang, "Single-machine scheduling problems with time and position dependent processing times", *Annals of Operations Research*, vol. 186, no. 1, pp. 345–356, 2011.

[12] D. Biskup, "Single-machine scheduling with learning considerations", *European Journal of Operational Research*, vol. 115, no. 1, pp. 173 –178, 1999, ISSN: 0377-2217. DOI: https://doi.org/10.1016/S0377-2217(98)00246-X. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S037722179800246X.

[13] J. N. Gupta and S. K. Gupta, "Single facility scheduling with nonlinear processing times", *Computers & Industrial Engineering*, vol. 14, no. 4, pp. 387–393, 1988.

[14] F. Matsunari, K. Ogo, T. Abe, and M. Asai, *Process time estimating apparatus*, US Patent 5,495,430, Feb. 1996.

[15] F. Jiang, Y. Jiang, H. Zhi, Y. Dong, H. Li, S. Ma, Y. Wang, Q. Dong, H. Shen, and Y. Wang, "Artificial intelligence in healthcare: Past, present and future", *Stroke and Vascular Neurology*, vol. 2, no. 4, pp. 230–243, 2017, ISSN: 2059-8688. DOI: 10.1136/svn-2017-000101. eprint: https://svn.bmj.com/content/2/4/230.full.pdf. [Online]. Available: https://svn.bmj.com/content/2/4/230.

[16] J. D. Spiegeleer, D. B. Madan, S. Reyners, and W. Schoutens, "Machine learning for quantitative finance: Fast derivative pricing, hedging and fitting", *Quantitative Finance*, vol. 18, no. 10, pp. 1635–1643, 2018. DOI: 10.1080/14697688.2018.1495335. eprint: https://doi.org/10.1080/14697688.2018.1495335. [Online]. Available: https://doi.org/10.1080/14697688.2018.1495335.

[17] C. Antoniou and H. N. Koutsopoulos, "Estimation of traffic dynamics models with machine-learning methods", *Transportation Research Record*, vol. 1965, no. 1, pp. 103–111, 2006. DOI: 10.1177/0361198106196500111. eprint: https://doi.org/10.1177/0361198106196500111. [Online]. Available: https://doi.org/10.1177/0361198106196500111.

[18] H. Kagermann, W. Wahlster, and J. Helbig, "Recommendations for implementing the strategic initiative industrie 4.0 – securing the future of german manufacturing industry", acatech – National Academy of Science and Engineering, München, Final Report of the Industrie 4.0 Working Group, 2013. [Online]. Available: http://forschungsunion.de/pdf/industrie_4_0_final_report.pdf.

[19] A. Diez-Olivan, J. Del Ser, D. Galar, and B. Sierra, "Data fusion and machine learning for industrial prognosis: Trends and perspectives towards industry 4.0", *Information Fusion*, vol. 50, pp. 92–111, 2019.

[20] S. Rawat and S. Rawat, "Multi-sensor data fusion by a hybrid methodology–a comparative study", *Computers in Industry*, vol. 75, pp. 27–34, 2016.

[21] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system", in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, ACM, 2016, pp. 785–794.

[22] J. Tukey, "W.(1977). exploratory data analysis", *Reading: Addison-Wesley*,

[23] B.-t. Zăbavă, G. Voicu, M.-N. Dinca, N. Ungureanu, and M. Ferdes, "Durability of pellets obtained from energy plants: Review", May 2018. DOI: 10.22616/ERDev2018.17.N419.

[24] Anyang Gemco Energy Machinery Co., Ltd., *A guide to small scale pellet mill*, http://www.pelletmillequipment.com/document/TechnicalGuide-on-Wood-Biomass-Pellets-Production-final.pdf, [Online; accessed 02-Aug-2019].

[25] D. H. Wolpert and W. G. Macready, "Coevolutionary free lunches", *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 6, pp. 721–735, 2005.

[26] S. Seabold and J. Perktold, "Statsmodels: Econometric and statistical modeling with python", in *9th Python in Science Conference*, 2010.

[27] P. J. Huber and E. M. Ronchetti, "Robust statistics john wiley & sons", *New York*, vol. 1, no. 1, 1981.

[28] D. R. Anderson and K. P. Burnham, "Commentary on models in ecology", *Bulletin of the Ecological Society of America*, vol. 82, pp. 160–161, Jan. 2001. DOI: 10.2307/20168551.

[29] K. P Burnham and D. R Anderson, *Model Selection and Multi-Model Inference: A Practical-Theoretical Approach*. Jan. 2002, vol. 2, p. 2. DOI: 10.1007/978-1-4757-2917-7.

[30] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, *Decision trees*, https://scikit-learn.org/stable/modules/tree.html, [Online; accessed 12-Aug-2019].

[31] L. Breiman, *Classification and Regression Trees*. Routledge, 1984. DOI: 10.1201/9781315139470.

[32] L. Rokach and O. Z. Maimon, *Data mining with decision trees: theory and applications*, ser. Machine Perception Artificial Intelligence. World scientific, 2014, vol. 81.

[33] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees", *Machine Learning*, vol. 63, pp. 3–42, Apr. 2006. DOI: 10.1007/s10994-006-6226-1.

[34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python", *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[35] J. H. Friedman, "Greedy function approximation: A gradient boosting machine", *Annals of statistics*, pp. 1189–1232, 2001.

[36] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions", *Machine Learning*, vol. 37, no. 3, pp. 297–336, 1999, ISSN: 1573-0565. DOI: 10.1023/A:1007614523901. [Online]. Available: https://doi.org/10.1023/A:1007614523901.

[37] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting", *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[38] T. Chen, *Introduction to boosted trees*, Slides available from https://homes.cs.washington.edu/~tqchen/pdf/BoostedTree.pdf, [Online; accessed 17-Aug-2019], Oct. 2014.

[39] T. Chen and C. Guestrin, *Xgboost: Extreme gradient boosting*, https://github.com/dmlc/xgboost, Research project at University of Washington [Online; accessed 20-Aug-2019], 2014.

[40] ——, *Xgboost parameters*, https://xgboost.readthedocs.io/en/latest/parameter.html, [Online; accessed 19-Aug-2019].

[41] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree", in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., Curran Associates, Inc., 2017, pp. 3146–3154. [Online]. Available: http://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree.pdf.

[42] T. R. Jensen and B. Toft, *Graph coloring problems*. John Wiley & Sons, 2011, vol. 39.

[43] J. Aarshay, *Complete machine learning guide to parameter tuning in gradient boosting (gbm) in python*, https://www.analyticsvidhya.com/blog/2016/02/complete-guide-parameter-tuning-gradient-boosting-gbm-python/, [Online; accessed 14-Aug-2019], 2016.

[44] D. Tarasov, A N. Medvedev, A. Sergeev, and A Buevich, "Review and possible development direction of the methods for modeling of soil pollutants spatial distribution", vol. 1863, Jul. 2017, p. 050 014. DOI: 10.1063/1.4992211.

[45] F Rosenblatt, "The perceptron–a perciving and recognizing automation", *Report 85-460-1 Cornell Aeronautical Laboratory, Ithaca, Tech. Rep.*, 1957.

[46] *Activation function — Wikipedia, the free encyclopedia*, https://en.wikipedia.org/wiki/Activation_function, [Online; accessed 26-Aug-2019], 2019.

[47] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines", in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[48] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models", in *Proc. icml*, vol. 30, 2013, p. 3.

[49] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning", *nature*, vol. 521, no. 7553, p. 436, 2015.

[50] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions", *arXiv preprint arXiv:1710.05941*, 2017.

[51] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, pp. 83, 149 –150, 170 –173, 200 –209, 290–294, 305–306, http://www.deeplearningbook.org.

[52] D. Masters and C. Luschi, "Revisiting small batch training for deep neural networks", *arXiv preprint arXiv:1804.07612*, 2018.

[53] B. Polyak, "Some methods of speeding up the convergence of iteration methods", *Ussr Computational Mathematics and Mathematical Physics*, vol. 4, pp. 1–17, Dec. 1964. DOI: 10.1016/0041-5553(64)90137-5.

[54] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization", *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[55] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization", *arXiv preprint arXiv:1412.6980*, 2014.

[56] Y. Dauphin, H. de Vries, and Y. Bengio, "Equilibrated adaptive learning rates for non-convex optimization", in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., Curran Associates, Inc., 2015, pp. 1504–1512. [Online]. Available: http://papers.nips.cc/paper/5870-equilibrated-adaptive-learning-rates-for-non-convex-optimization.pdf.

[57] S. Ruder, "An overview of gradient descent optimization algorithms", *arXiv preprint arXiv:1609.04747*, 2016.

[58] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning", in *International conference on machine learning*, 2013, pp. 1139–1147.

[59] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting", *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html.

[60] G. E. Hinton, A. Krizhevsky, I. Sutskever, and N. Srivastva, *System and method for addressing overfitting in a neural network*, US Patent 9,406,017, 2016.

[61] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification", in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3642–3649. DOI: 10.1109/CVPR.2012.6248110.

[62] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[63] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups", *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012, ISSN: 1053-5888. DOI: 10.1109/MSP.2012.2205597.

[64] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop", in *Neural networks: Tricks of the trade*, Springer, 2012, pp. 9–48.

[65] L. Breiman, "Random forests", *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[66] A. Fisher, C. Rudin, and F. Dominici, "All models are wrong but many are useful: Variable importance for black-box, proprietary, or misspecified prediction models, using model class reliance", *arXiv preprint arXiv:1801.01489*, 2018.

[67] C. Molnar, *Interpretable machine learning: A guide for making black box models explainable*. Christoph Molnar, 2019, `https://christophm.github.io/interpretable-ml-book`, Section 5.5.

A Additional figures

HA1_TOEV

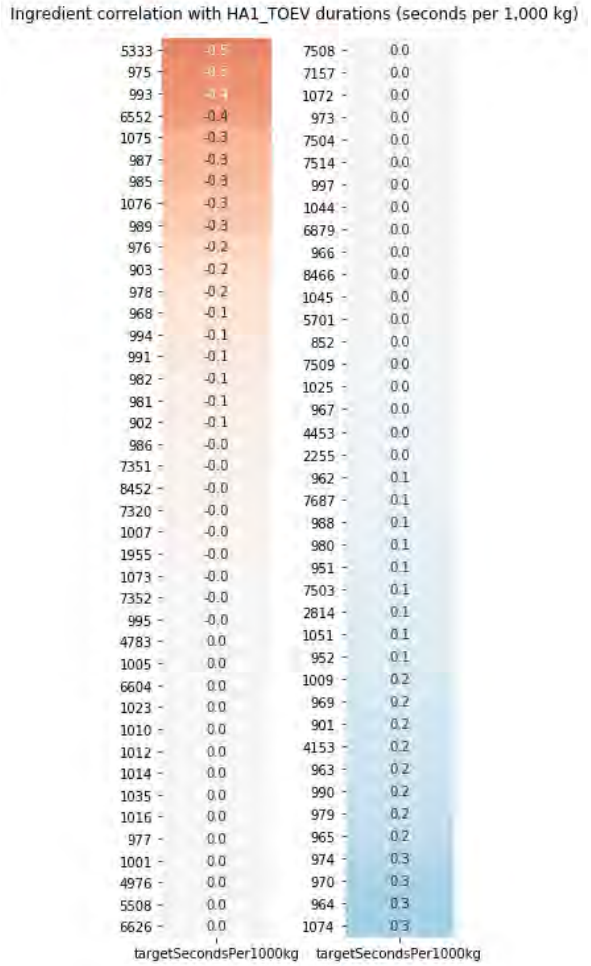


FIGURE A.1: Pearson correlation between ingredient percentage features and HA1_TOEV durations (targetSeconds).

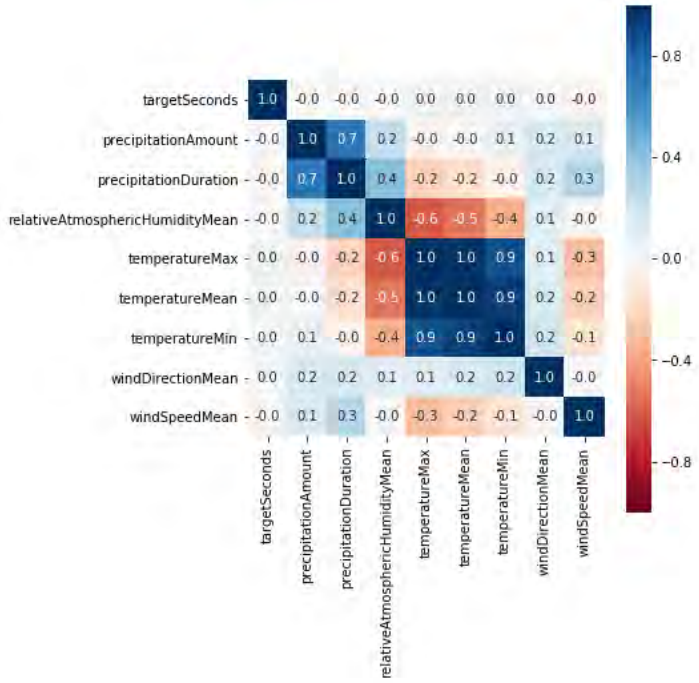


FIGURE A.2: Pearson correlation between weather features and HA1_TOEV durations (targetSeconds).

NM1_step

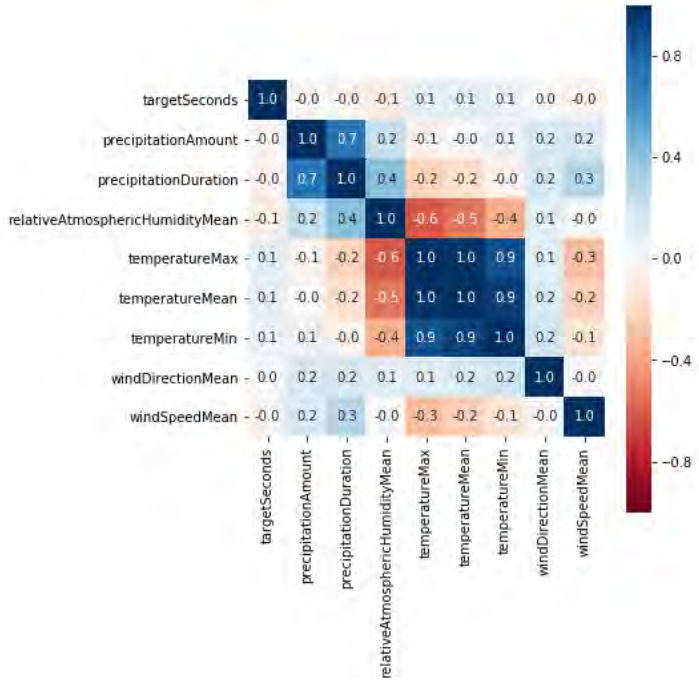


FIGURE A.3: Pearson correlation between features and NM1_step durations (targetSeconds).

MML_AFV_TR

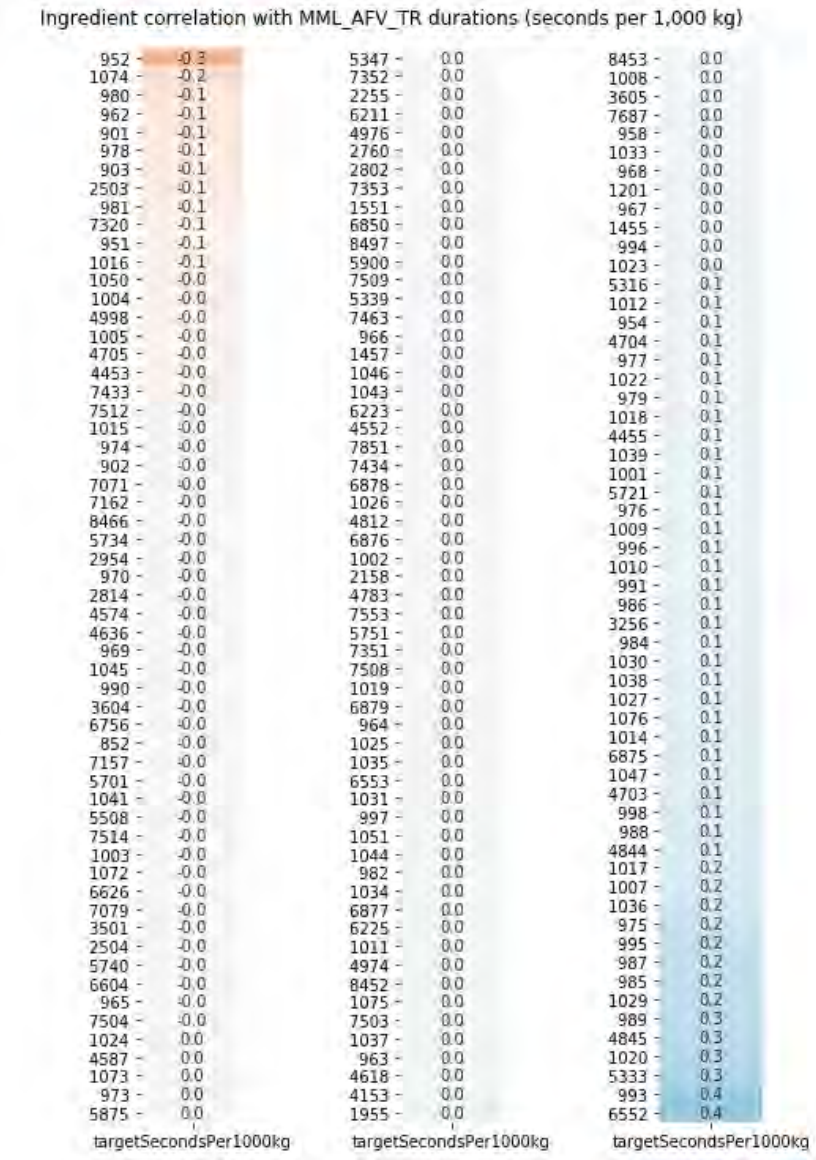


FIGURE A.4: Pearson correlations of ingredient percentage features and MML_AFV_TR durations.

PL_PO1_slope

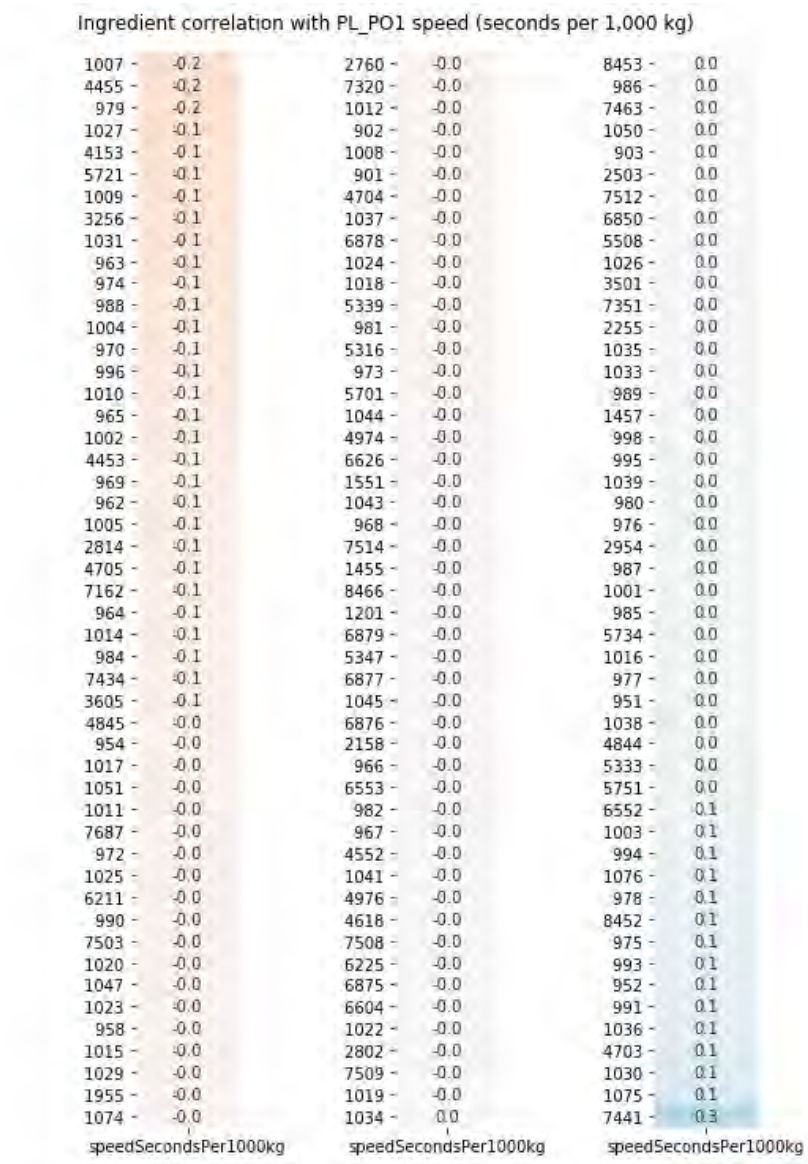


FIGURE A.5: Pearson correlation between PL_PO1_slope values and ingredient percentage features.

PL_PO4_slope

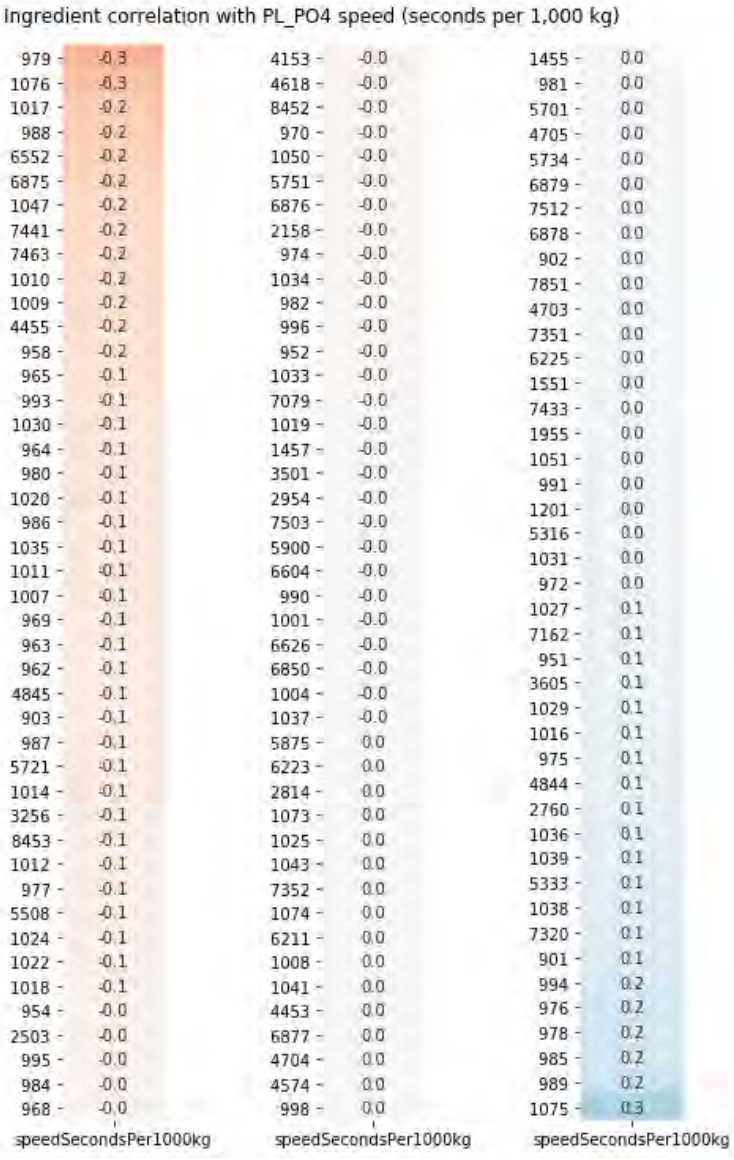


FIGURE A.6: Pearson correlation between PL_PO4_slope values and ingredient percentage features.

PL_PO4

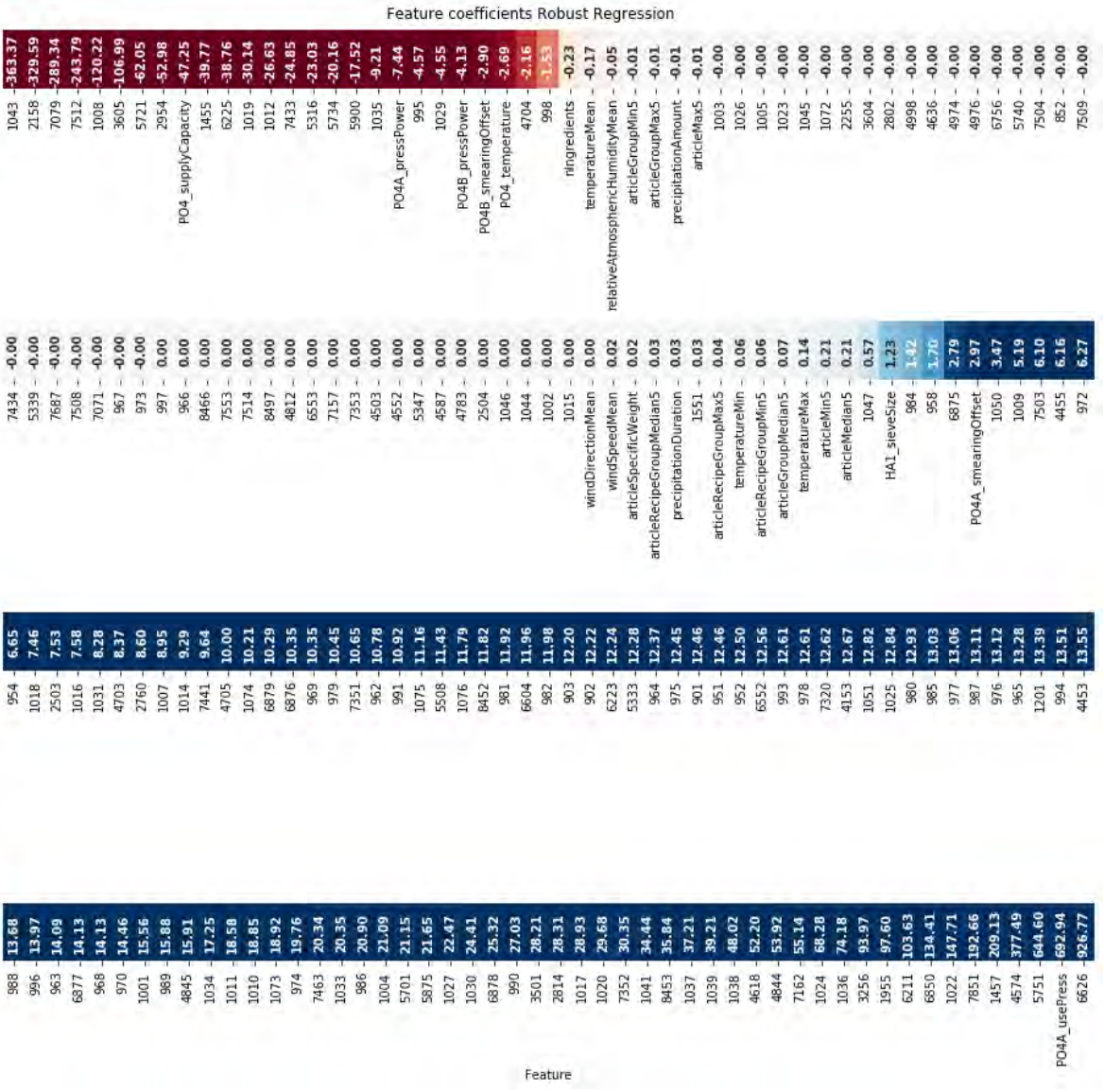


FIGURE A.7: Feature coefficients of Robust Regression for the estimation of the PL_PO4 durations.