



Optimizing Control of a production line using Machine Learning

Optimizing Control of a production line using Machine Learning

by
Koen Kroep
k.kroep@student.utwente.nl
s2280019
October 15, 2024

This thesis is written as graduation assignment for the study Industrial Engineering and Management at the University of Twente.

1st Supervisor:
Prof.dr.ir. Martijn Mes
University of Twente

2nd Supervisor:
dr. Engin Topan
University of Twente

Company Supervisor:
Menno Hoeksema
ErgoDesign B.V.

University of Twente
Post office Box 217
7500 AE Enschede

Ergodesign B.V.
Auke Vleerstraat 22
7521 PG Enschede

Faculty of Behavioural, Management and Social Sciences
Master Industrial Engineering and Management

Preface

With the completion of this thesis I will finish my study time at the University of Twente. This thesis is written as part of the graduation assignment in the master Industrial Engineering and Management.

I would like to thank my supervisors at ErgoDesign, Menno Hoeksema, Joey Klein Koerkamp and Dimitry Brons, who gave me the opportunity to complete my graduation assignment with their supervision and guidance.

I also would like to thank Martijn Mes and Engin Topan from the University of Twente, who guided me through this process and provided helpful feedback. This supported me in the process of finishing my studies.

I also would like to thank my family and friends, who have supported me during my time studying in Enschede, which has helped me to complete my studies.

Kind regards,

Koen Kroep

Enschede, October 2024

Management Summary

The main research problem in this thesis is that the optimal speed and the buffer levels when machines should be switched on and off are unknown. The operators of the production line use speed to control the buffer level and aim to produce as much as possible. The buffer levels at which the machines can be switched on and off help with this objective, because they decide what machines to stop to increase production speed of high prioritized products.

The production line central in this thesis is a packaging line where machines are packaging chocolate products. There are in total six machines, and only the last machine has a buffer. Products that could not be packaged in the end are wasted. Along the production line are machine failures, which have stochastic moments and duration.

Operators take both decisions with as goal to produce as much products as possible. However, when speed is too high, and with machines failing, the buffer might fill up. As soon as the buffer is full, products can be produced to waste. This is also the case when too many machines are switched off. When the speed is too low, the production quantity is not maximized, which means that more products could have been produced.

At the moment, the buffer levels to decide what machines to switch on and off are summarized in nine “scenarios”, which are pre-determined in advance. The scenario is chosen based on which product type, and therefore which machine, is prioritized. The scenario is determined in advance of production, the speed can be controlled by the operators during production. However, the speed can range between ■■■ and ■■■ cuts per minute. Speed can be changed in the processing department and is only noticed in the packaging department after 20 minutes.

Machine learning techniques can help to predict production output, in terms of amount of products produced and wasted. However, to train this on a real life environment takes a lot of time, and decisions made there are not always optimal, which may lead to increased amount of wasted products. However taking these worse are needed to learn these effects on the production line.

To overcome this issue, a simulation model can be used, which serves as replica of the real life production line. This model helps in the creation of a dataset for training purposes of the machine learning algorithm.

Training requires selection of different input samples. A sample is the real life state of the production line at a certain point in time. Because of the simulation model, all possible input scenarios could be selected and used to train the machine learning models. However, the large amount of possibilities require that a selection should be made, to reduce the training time. Per sample, multiple actions can be taken regarding speed and the buffer level when machines can switch on and off. The model should decide which action and sample to run using the simulation model, in order to observe the output value. This output value is a the difference between production and waste, where each component has its own weight.

A training dataset can be obtained by using a simulation model. This simulation model will

speed up the collection of data. With this dataset, a machine learning algorithm is trained to predict this output.

The use of simulation improves the training time needed to create input data for the model. Less observations are needed from the real production line, meaning the real production process is not bothered when in the training stage. The results of the simulation model suffer from large variation in the output. To overcome these issues, more simulation observations are combined per sample. The mean objective value is calculated after 10 simulation runs, and given to the machine learning model during training. This reduces the mean squared error and improves the prediction accuracy of the ML model.

During validation, we notice that both machine learning models are able to predict the correct speed, and follow the same pattern per predicted speed on two given input samples. The models show that they are able to predict the correct speed on the given samples. However, when calculating the mean squared error of both models, the Gradient Boosted Decision Trees work better than the Neural Network. Also when we compare the running time, the gradient boosted decision trees perform better in creating a fast advice, within a second, while the neural network takes 10 seconds.

In case also the pre-determined line settings are removed, and are added to the agents' decisions, we notice that both the simulation model as well as the machine learning algorithm suffer from small deviations in the results per action. This means that the model is not trained well enough to be able to compare these different actions to a reliable advice.

However, when the model only advises on speed, the performance of the model is sufficient to provide correct advice to the operators in 86% of the cases for the best working model. Before implementation, a period should be used where the machine learning model and production line are simultaneously running. Advice can then be calculated, and implemented by operators, or rejected. When rejected, a reason can be logged to see how the model can be improved in the future.

Table 1 shows the characteristics of the best performing model. The model with best performance is the Gradient boosted decision trees. Its Mean squared error is 0.0013, and in 86% of the samples during validation it predicted the optimal speed correctly. This means that this model can help operators to take decisions regarding speed of the production line.

Aspect	Performance
Prediction model	Gradient boosted decision trees
Interval between advices	30 minutes
MSE	0.0013
Prediction correctness	86%

Table 1: Best model performance

Concluding, we created a model that provides operators with validated advice, where the prediction is on average close to the real production output, as measured by the simulation model. However, this advice is created within a second in the case of the gradient boosted decision trees, instead of the long duration needed by the simulation model.

Contents

1	Introduction	1
1.1	Problem Introduction	1
1.2	Problem identification	2
1.3	Research Design	5
1.3.1	Research questions	5
1.3.2	Plan of Approach	7
1.3.3	Scope	7
1.3.4	Deliverables	7
1.4	Outline	7
2	Context Analysis	9
2.1	Production line layout	9
2.2	Production schedule	11
2.3	Machine Standstills	13
2.3.1	Machine failure duration	15
2.3.2	Changeovers	15
2.4	Analysis of the buffer	16
2.5	Decision variables	17
2.6	Conclusion	18
3	Literature	20
3.1	Background	20
3.1.1	Simulation Optimization	21
3.1.2	Ranking and Selection	21
3.1.3	Multi-armed bandits	21
3.2	Production line optimization	21
3.3	Digital Twin	22
3.4	Machine Learning	23
3.4.1	Reinforcement learning	24
3.4.2	Supervised Machine Learning	24
3.4.3	Model evaluation	25
3.5	Machine learning algorithms	25
3.5.1	Gradient boosted decision trees	25
3.5.2	Artificial Neural Networks	26
3.6	Conclusion	27
4	Solution method	29
4.1	Training	29
4.1.1	Contextual bandit	30
4.1.2	Actions	33

4.2	Testing	34
4.3	Using	34
4.4	Conclusion	35
5	Simulation modelling	36
5.1	Modelling assumptions	37
5.2	Input	37
5.3	Production process	38
5.4	Objective	40
5.5	Conclusion	41
6	Machine learning algorithm	42
6.1	Data analysis	42
6.2	Data preparation	43
6.3	Cross-validation	43
6.4	Prediction functions	44
6.4.1	Gradient boosted decision trees	44
6.4.2	Neural Network	45
6.5	Conclusion	45
7	Experiments	47
7.1	Experimental setup	47
7.1.1	Gradient boosted decision trees	48
7.1.2	Neural network	49
7.2	Results	49
7.2.1	Machine Learning algorithm performance	49
7.2.2	Sample selection	50
7.2.3	Simulation	52
7.2.4	Advice creation	54
7.3	Validation	55
7.4	Conclusion	60
8	Conclusion and Recommendations	62
8.1	Conclusion	62
8.2	Recommendations	63
8.3	Limitations and future research	64
A	Machine failures	70
B	Technical simulation description	71
B.1	Initialization	71
B.2	Process	71
B.3	Statistics	72
C	Subscenarios	73
D	Simulation results	75

Reader's guide

This readers guide provides an overview of what can be read in each chapter.

Chapter 1: In Chapter 1 we provide an introduction to the problem that is central in this thesis. We provide the research questions and plan of approach to solve the problem.

Chapter 2: In this chapter we provide an overview of the current situation, how the current packaging process works and what the problem is in terms of determination of speed of the packaging line.

Chapter 3: This chapter describes what is known in literature about control of an automated production line, and how machine learning or artificial intelligence can help improving the performance of the production line. This chapter describes some methods regarding this automation, production and control, reinforcement learning and contextual, multi-armed bandits.

Chapter 4: In this chapter, we describe the methods of selecting samples with which the machine learning algorithms will be trained.

Chapter 5: In this chapter we describe the simulation model, explain the working of the simulation model and what assumptions are made to create an accurate model. We describe the handling of products and selection of statistics.

Chapter 6: In this chapter, we describe the data preparation for the machine learning models, we describe which models to select and we describe how to train these models effectively.

Chapter 7: This chapter describes the experiments that are performed using several methods, and describes the results of these experiments.

Chapter 8: In this chapter, we conclude the performance of all models, give a recommendation of implementation, describe the limitations and finally discuss methods of further research.

Appendix: In the appendix, an overview of machine failures can be found. Afterwards, we describe the technical implementation of the simulation model in Plant Simulation and describe the different sub-scenarios or linesettings that are used in the current situation.

List of abbreviations

AI	Artificial Intelligence
ML	Machine Learning
RL	Reinforcement Learning
DL	Deep Learning
DRL	Deep reinforcement learning
ABM	Agent based modelling
ABMS	Agent based modelling system
PPC	Production planning and control
KPI	Key performance indicator
(D)NN	(Deep) Neural Network
DT	Digital Twin
AI-DTS	Artificial Intelligence - Digital Twin System
NN	Neural Network
GBDT	Gradient boosted decision tree

Chapter 1

Introduction

In this chapter, we introduce the company and the research goal. We identify the action problem and core problem according to the procedure by Heerkens and Winden, 2017. Besides, we construct the research questions and plan of approach of this research. In the end, this chapter provides the scope and deliverables of this project.

Nowadays, real time decision making is becoming more and more important, as decisions regarding production processes are required to be taken in reaction to a frequently changing environment. Standard algorithms and methods, such as simulation and planning and scheduling, do not cover this fast decision making (Tremblay et al., 2024). Therefore, new evolution's in machine learning become more important in the decision making processes at different production companies.

ErgoDesign B.V., an Industrial Engineering company located in Enschede, the Netherlands, provides advice to different companies on their production processes. They operate in multiple sectors, such as logistics by means of improving efficiency, smart scheduling and simulation. Because of the ever-changing nature of production processes, a model or algorithm should be able to provide instant advice to companies on their production process layout, design and on what decisions should be made to operate the production line efficiently, such as production speed.

First, Section 1.1 introduces and describes the problem that is central in this thesis. Then, Section 1.2 identifies causes and relationships between underlying problems, resulting in the formulation of a core problem. In Section 1.3, the research questions, plan of approach, scope and deliverables

1.1 Problem Introduction

This section introduces the main problem and defines several causes. We construct a problem cluster (Heerkens and Winden, 2017) to identify the causal relationships between these problems and eventually define a core problem. We select the core problem as the eventual cause of the action problem, and the following plan of approach describes how to solve this core problem.

A company, operating a production line producing candy bars, is facing challenges with the adjustment of the production line to the real-time changing production environment. These changes in the production environment are mainly caused by machine failures. This production line runs through two departments: processing and packaging. The processing department produces the products. The conveyor belt then transports the products to the packaging department where these products are packed before being shipped to a customer.

The packaging department consists of one production line, with multiple packaging machines. These machines take products from the conveyor belt to pack these products. The buffer, placed just in front of the last machine, catches all products that are not taken by any machine before. However, if this buffer is full, products fall off the line and are wasted. The last machine on the production line packages all products that are taken by the buffer.

The company aims to produce as many products as possible, however while keeping the waste at an equal size. The company can increase production by adjusting the speed of the production line at which products flow over the conveyor belt and adjusting the state of the machines. Not all machines are running at the same time. Machines can be stopped to enable other machines to catch more products. This is done when some products have a higher priority than others. Machines that produce the products with lower priority can then be stopped.

There are multiple methods available for operators along the line to increase production output. One of the factors that can be changed is the speed of the production line. This speed is measured as cuts per minute made by the knife at the beginning of the conveyor belt. By decreasing this speed, fewer products enter the packaging department, while increasing the speed results in higher production.

The other method is the working of the machines. As soon as the buffer fills up, certain machines that are stopped, due to lower priority, can be set to work again. This results in a higher number of product rows that can be taken by any of the machines, therefore decreasing the fill rate of the buffer. However, this also means that products that are to be picked by machines later in the line are getting fewer products. This results in a delay in reaching the goal for a certain product type and is therefore not always desirable.

1.2 Problem identification

The factory faces difficulty in determining the optimal speed and machine settings of their production line. This results in a high number of production waste, due to the unavailability of machines and buffers to handle these products. However, there is no optimal number of products as output due to disruptions in the production, or no optimal chosen speed and machine settings. The availability of machines differ throughout the time, meaning that at one instance, many products are wasted, and at another instance of time, more products can be produced. This leads to increasing costs and work for personnel and a longer time to process the needed amount of products than actually needed. Some products have finished production, but cannot be processed at the packaging machines, and are therefore lost as waste. On the other hand, a higher amount of products can be produced when correctly adjusting the speed and machine settings.

The buffer aims to catch products that other machines have not taken earlier in the process and prevent them from becoming waste. The fact that other machines are not taking any products is either due to failures of these machines or these machines already picked a row of other products, and are therefore already processing items.

However, it can also be due to the settings of the production line. When some machines have lower priority, these machines stop producing or produce at a slower pace, to give priority to other machines, which use different packing materials and therefore deliver different products. The final machine picks products that are left on the production line if there is space in the buffer.

When the speed of the production line is too high, a larger amount of products end up unprocessed at the end of the production line as lost products, while a too-low speed means that machines will produce fewer items than the capacity of the production line. Besides, a change in

the operating status of the different machines across the production line influences the fill rate of the buffer and the amount of products wasted at the end. Therefore, the production line needs a balance to improve the output, while maintaining the current level of waste, by adjusting the configurations.

The action problem (Heerkens and Winden, 2017) therefore is that at the moment there is a high number of lost products, and no optimal configurations are used at the production line. The amount of waste should not increase, but the output should be as high as possible. The description of the action problem is given below.

Action problem: The output of produced candy bars at the production line is not maximized.

This action problem has several causes, which itself are caused by other factors. To visualize these causal relationships, we construct the problem cluster, as can be seen in Figure 1.1. At the top, the action problem is given in two boxes, as they are two individual problems and are happening simultaneously. They are influenced by other factors as well. We define all causes of this action problem below, resulting in a final cause, the core problem. This core problem will be the main problem to solve during the remainder of this thesis. The action problem is suffering at one side from waste of production, caused by wrong settings in the production line. On the other side, the output of the production line can be increased in certain situations.

At several moments, the output of the production line is too low. The production line misses a certain balance. This balance should aim to keep the current amount of waste equal while optimizing production output. As these are two different problems, that are both noticed in the production process, we consider the problems as independent, and add them separately to the problem cluster in Figure 1.1.

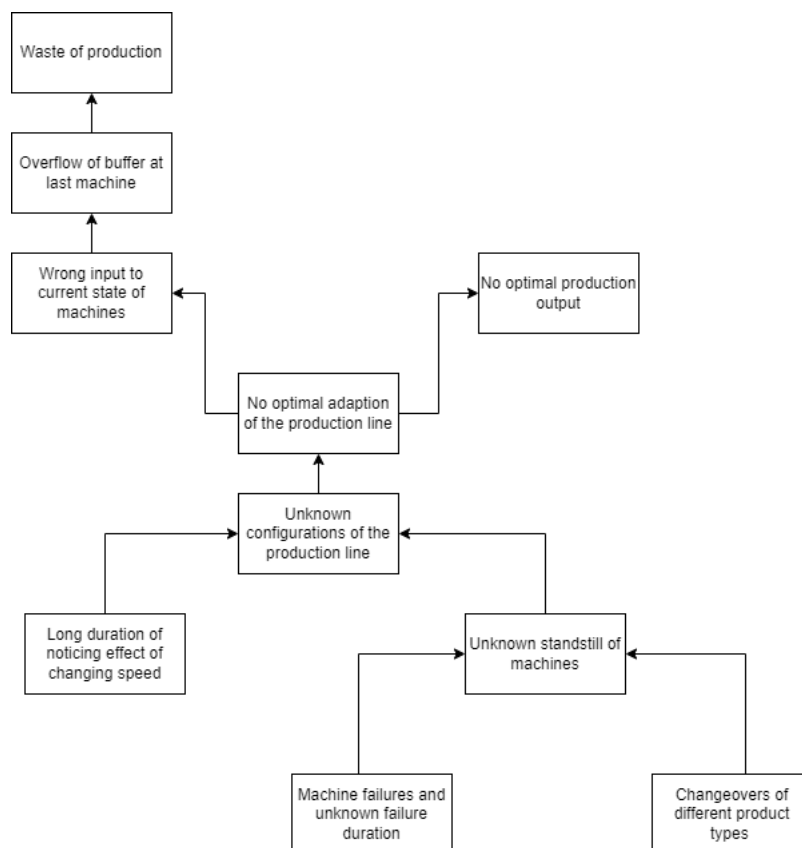


Figure 1.1: Problem Cluster

In the problem cluster in Figure 1.1, the first part of the action problem is the waste of products. This means, that of all products, there is waste that ends up at the end of the line. As these are perfect products that could have been sold, this should not be too high, but production should keep it at a certain threshold. The main cause of this product waste is the overflow of the buffer. This buffer is located just before the last machine. This means that each product that comes by this buffer and the machine therefore does not take, is lost as scrap and is not sold.

However, the products also go along the production line across other machines, which were unable to take the products and process them. This could have as a reason that there is already another product in that machine or the machine is failing, or is at lower priority, and therefore switched off. To summarize these reasons, the main problem is that there is a wrong input in the current state of the machines along the production line, which means that the machines were unable to pick all the products, resulting eventually in waste.

Another part of the action problem is the fact that there is no optimal production output, or it is unknown if the optimal production output has been reached. This means that the production line could have processed more products, while also keeping the amount of waste in products the same. Production is done according to a specific schedule, which is predetermined. This schedule provides what machines should be producing and how many products.

For both the wrong input due to the current state of machines and the fact that there is no optimal production output, we assign one factor as cause. This factor is that there is no optimal input as a production scheme for the production line. Machines are currently standing still due to certain, predetermined scenarios. This production scheme determines which machines are working and which are standing still. Also, the production scheme determines the speed of each packaging machine. This production scheme therefore has a major impact on the output that is produced at the packaging line.

For an optimal output, the speed at which products are being processed should be as high as possible, and as many machines as possible should be working, while to match the input of the production line with the availability of the machines, the speed should be reduced. Therefore, there is no optimal speed reached at the moment, what results in the occurrence of the action problem. However, when there is a need for a specific type of product, some machines should stop their production, to prioritize other machines. These prioritized machines should be producing. However, when the buffer is reaching its capacity, it is desirable to have machines that were stopped producing again, in order to prevent the buffer from completely filling up and producing waste.

It is at the moment unknown what the optimal settings for speed and state of machines are at a specific moment in time. This depends on several factors, with as cause that it is unknown what speed should be optimal, when to change the production line to that speed, and what machines should be running or standing still. This is caused by the fact that there is a long duration before the change in speed is seen in the production line.

However, besides this duration in change, it is unknown when machines will fail, and at what period in time they will be operative again. This determines to what extent the speed needs to be changed, as well as the time when this decision needs to be made. Also, because of the long duration of the effect of a speed change, not at every machine failure, a new decision needs to be made. Therefore, there is at the moment no knowledge about the optimal time that a system change should be made.

Multiple machines can work on different items. This means that several times, the production line requires changeovers, to enable other production types. This results as well in a standstill of a machine. The main part of the schedule is known, but the exact times when a changeover takes place is depending on when the desired production goal is reached for a type of product. This is

uncertain, due to machine failures. Each item has a certain goal of how many products should be produced, according to demand and capacity, and when this goal is reached, a machine will change to another product type. However, due to uncertainty in machine failures, it is unknown when a machine exactly reaches this goal. Therefore, the time when a changeover will take place is also unknown.

The changeover and uncertainty in machine failures and duration are related to the uncertainty of the standstill of machines. The machines will keep failing, with duration and moment uncertain, and changeovers are always necessary. Due to these combinations, there will always remain uncertainty in the moment and duration of a machine that is standing still. Also, the fact that the change in speed is only seen after a certain period is not changeable, because of the layout of the production line. These factors are taken as input factors in this research, and we take them into account as important properties for an eventual model.

In the process is much uncertainty. For example, the failure of the machines is stochastic. A speed decision should be made very fast, to prevent products from becoming waste and keep the output of the production line high. This means that a model should be constructed that determines the output as speed and which machines should have priority of production, given different states of the line, which depends on the availability of machines, state of the buffer, current speed, and fulfilment level of goals per product item. The changes in variables throughout the production line are frequent. This has an influence on the state of the buffer, which should be taken into account in all decisions. Thus, we identify the core problem as follows:

Core problem: There is no policy to control the production speed depending on the current state of the production line.

A policy should be created which determines the optimal speed to run at the production line. By solving this core problem, a model or tool should provide an improved policy about the optimal configurations of the production line at every point of time in which a decision should be made, given the current state of the production line. These optimal input configurations should take the balance of the action problem into account, where there is an optimal balance between production output and keeping the waste equal to the current situation.

1.3 Research Design

In this section, we describe a plan of approach about how to solve the core problem which is identified above. Also, we present the research questions for this research project. The research design followed in this thesis is the design as described by Heerkens and Winden, 2017.

The core problem, that there is no policy about the speed the production line should have, should be solved by creating a model, that would automatically advise about the best configurations, based on available historical data. These decisions need to be fast, to adapt to the frequently changing environment of a production line. The goal therefore is to create a model that could make these decisions about optimal configurations of the processing line. It should take care of all different considerations that play a role in the number of output and waste of the line. This Section has the following structure. In Section 1.3.1 we construct the research questions. Section 1.3.2 describes a way of solving the research questions. Section 1.3.3 describes the scope and Section 1.3.4 the deliverables of this research.

1.3.1 Research questions

The research questions provide a structured way of finding a solution to the core problem. Each research question comes with several sub-questions, which partly aim to answer the research question. The main research question that is central to this research is as follows:

How can decisions regarding speed of a packaging production line optimize the output of production while the waste does not increase?

The following sub-questions aim to provide a structured approach to answer the main question.

1. *How does the current production process work?*
 - What is the current process of the production line?
 - How is the production schedule constructed?
 - How are decisions in changes of production taken?
 - What is the current output of the production line?
2. *Resulting from a literature study, what solution methods exist to optimize and automate the configurations of the production line?*
 - What methods do exist to optimize the output of a production line given uncertainty?
 - What are the possibilities to intervene in production speed to optimize output?
 - How can Machine Learning contribute to the optimization of a production process?
3. *How can a tool be designed that is able to provide advice about the production line?*
 - What model does fit best with the available data?
 - What ranges of input should be considered within the model?
 - How can a set of training data be designed?
4. *How can a simulation model be designed that evaluates the performance of the production line?*
 - What assumptions are made during simulation modeling?
 - What type of simulation model is needed?
 - What factors should be considered in the simulation model?
5. *How can production output and waste be predicted?*
 - How can functions be trained to predict output?
 - How can decisions be made from the prediction?
6. *How does the Machine Learning model perform?*
 - How does training influence the performance of the model?
 - Can parameters be changed to increase performance?
 - What is the effect of the model on measured output?
 - How much impact does the model have on the production of waste?
7. *What recommendations and conclusions can be made from the model and the experiments?*
 - What is the impact of the model to the processors across the line?
 - How can the model be used alongside a production line?

1.3.2 Plan of Approach

This subsection gives a plan of approach to answer the research questions as described in Section 1.3.1. To solve the action problem, first, the current situation is analyzed. The problems that arise from the action problem resulting from the current way of working are analyzed, and with that, the main causes for the core problem will become visible. Data analysis will be performed to see the current amount of lost products at the current speed and will identify the status of the machines and the buffer. A model will be constructed that should give, given the current state of the machines, buffer, and speed of the production line, an optimal decision for configurations on the production line, while not increasing the amount of produced waste.

1.3.3 Scope

The scope of this research is based only on the last part of the production line. The part of the production line before the packaging line is not considered in this research. It is assumed that the processing line before will never be a problem within this process, and the production will give no constricted amount of supply for the machines while the production line is running. Figure 1.2 shows the scope of this research visually.

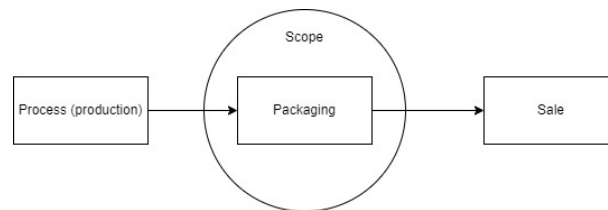


Figure 1.2: Scope

1.3.4 Deliverables

The deliverables of this research are as follows:

- Model to create an advice about speed and change in production configuration.
- Balanced advice about speed and production configuration, considering production output and waste. This decision can be about the speed, but also about the type of schedule and what products to produce.
- Thesis to explain and elaborate on the deliverables above.

1.4 Outline

In this chapter, first, we described the problem identification, where we identified the action and core problem. Research questions are constructed in order to provide a structured methodology of solving this core problem. Also, the deliverables and a plan of approach how to solve the core problem are given. The rest of this thesis has the following outline. In Chapter 2 an overview is given of the production line, the relevant input and output of this packaging line and a short data analysis of the machine failures. Also, the production process is described. In Chapter 3 a first overview is given about what can be found in literature about this topic and research questions. Chapter 4 provides an overview of the solution method and sampling strategy. Chapter 5 first describes the simulation model, used to generate data. In Chapter 6 we construct the machine learning algorithms. In Chapter 7 we test the model with a training dataset and a testing set, to see the performance on several Key Performance Indicators of the different models, and optimize the machine learning algorithm by conducting different experiments. Afterwards,

1.4. OUTLINE

conclusions and recommendations are given, together with a plan on implementation within a real-life production line.

Chapter 2

Context Analysis

This chapter describes the current situation of decision-making by the employees along the production line, the operators. It provides an analysis of the data on the current functioning of the production line. We explain the production process. This chapter answers the following set of research questions, and the attached sub-questions as described in Section 1.3:

1. How does the current production process work?
 - What is the current process of the production line?
 - How is the production schedule constructed?
 - How are decisions in changes of production taken?
 - What is the current output of the production line?

This chapter aims to provide an overview of how decisions are made in the current situation, and how this influences the performance of the production line. Therefore, the different decisions that are made, and the different processes that are important regarding the output of the production line are analysed. Section 2.1 describes the process of the production line and its logic towards the packaging department. Section 2.2 describes the schedule of different products among the different machines. Section 2.3 describes the behaviour of the machines and gives a data analysis of the different types of failures. Section 2.4 provides insight into the behaviour of the buffer level regarding different other settings. Section 2.5 then provides the decisions made by the operators and the influence of machine failures and buffer level on the production line's output.

2.1 Production line layout

The production line central in this thesis is a production line in a factory for the packaging of candy bars. The packaging department is located after the processing department. This processing department produces and transports the products in rows over the conveyor belt to the packaging department. When they arrive there, the products are ready to be packed.

The packaging line consists of different machines along a conveyor belt. The conveyor belt runs at a certain speed. This speed is an important factor for the total output of the production line. The speed of the conveyor belt is measured in cuts per minute that the knife makes at the entrance of the packaging line. This knife cuts a long line of semi-finished products into the separate product bars. This speed determines the number of product rows entering the packaging department per minute. Figure 2.1 shows the conceptual representation of the packaging line.

In the packaging department, the machines pick the products from the packaging line according

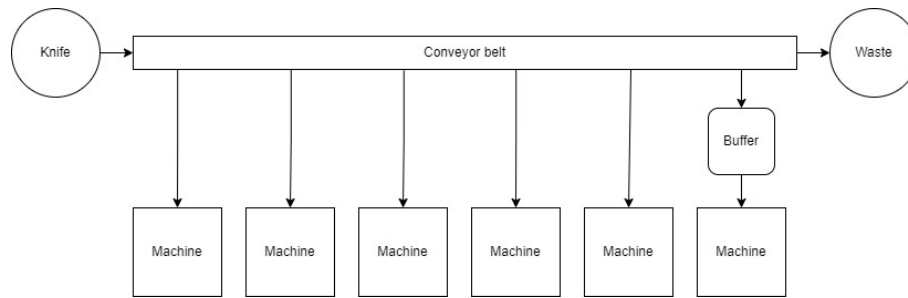


Figure 2.1: Conceptual representation of production line

to some logic. Figure 2.2 shows a flowchart of the logic of processing the bars in this processing line up to the last moment before actual packaging the products. This flowchart describes the process of how the machines select products for packaging after the products have entered the packaging part of the production line.

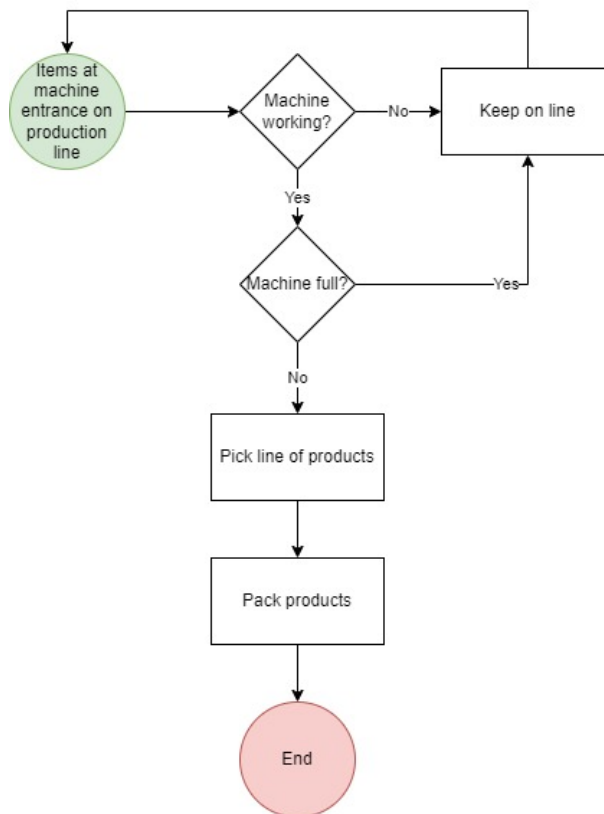


Figure 2.2: Flowchart of product selection at packing machines

Figure 2.2 shows the logic of how the machines select rows of products. At the packaging section of the production line, the chocolate bars enter in rows of 48 products, at a speed determined by the knife at the entrance. Machines take these rows when this machine is available. A machine is available for packaging a new row of products if the machine is both working, and the packaging machine is empty, so no products are being packed in that machine at the moment. If a machine is not available, the products continue on the conveyor belt, to be taken by another machine later on the production line. A packaging machine can only process one row of bars at the same time. Just in front of the last machine is a buffer, which catches off the remaining product rows. This buffer can take up to 385 rows of products in total. When this buffer is filled, all products that are not taken by any other machine previously in the process are thrown away as waste.

Some parts of the waste are reproducible, while other parts are not.

There can be several reasons why a machine is not taking products from the production line, besides when it is already packing other products. A machine standstill can either be planned or unplanned. A planned machine stop can happen due to maintenance, or there is a scenario where specific product types are prioritized. If a machine is not making these product types with a high priority, and other machines along the line are, then a machine can be set at a slower pace or is even stopped. However, this is depending on the fill level of the buffer. This is further explained in Section 2.2. Another option of why a machine is not operating is that the machine stops producing, but unplanned. In this case, there is a failure occurring in the machine. This means that the machine should be repaired. Section 2.3 analyses the behaviour of these machine failures.

The different machines along the production line can work with different packaging formats. Not all machines can produce all the different types of products. The products are in principle the same, but the packaging is different. This leads to the fact that a control method should provide a policy about which machines should operate and which should be at a standstill or run slower. This should provide a desired distribution of all different product packaging formats. If all machines run at the highest pace, the last machines receive fewer products than the first. This is not a desired situation when one of the last machines is producing packaging formats with higher priority.

Table 2.1 shows the different types of machines and which product type can be packaged on which machine. There are a lot of products that can only be produced on one single machine, and several different products that can be produced on three machines. Machines 2, 4 and 6 are therefore the same in terms of what they can produce. The company created scenarios to determine the product type that a machine is producing and at which pace. The planning department determines which scenario to run, based on, among others, demand forecasts and current stock levels. Section 2.2 explains these scenarios.

Product type	1	2	3	4	5	6	7	8	9	10	11	12	13
Machine 1					X	X			X	X			
Machine 2	X	X	X	X									
Machine 3							X	X	X	X	X		X
Machine 4	X	X	X	X									
Machine 5											X	X	X
Machine 6	X	X	X	X									

Table 2.1: Production type per machine

2.2 Production schedule

The production line operates according to a specific production schedule, also referred to as a scenario. The set of scenarios provides an overview of which machines should be working at what moment, depending on among others the buffer level, and which packaging formats the machines should produce. This is to make sure that for each product type and packaging type, a certain specified goal is achieved. This means, that in this production schedule, machines can be stopped to make sure that other machines will get sufficient products. However, this should not lead to an overflow of the buffer due to a standstill of too many machines. The production schedule therefore have to provide a moment that the different machines should be working again. However, this can be uncertain due to machine failures and changes in the speed of the production line.

The production schedule is based on 9 different scenarios. Each scenario determines which machines are switched off when this is allowed by the buffer level. The scenarios are used to make sure that the desired quantity of required products is produced. Scenarios enable in advance to give priority to specific product types, making sure that these are produced faster. This can be done by switching machines off. However, in all cases, the level of the buffer should be taken into account. This means that when the buffer is almost full, the scenarios also take into account that machines should be operating again.

Each scenario is divided into different sub-scenarios. These sub-scenarios provide an overview of when machines are switched on or off. These sub-scenarios are determined based on the fill level of the buffer. To provide a method to prevent this buffer from filling up too much, the speed and availability of the machines are adjusted within these scenarios. When the buffer level is too high, machines that were on stand-by are switched on again. The four divisions of the scenarios are mentioned below.

- Fill buffer: the buffer is empty and can be filled with products. Mainly the situation where several machines are on standby.
- Buffer normal: buffer reached a certain level, no need for changes.
- Empty buffer: buffer is starting to reach its maximum capacity, so need to use more capacity of machines to prevent this overflow.
- Empty buffer max: buffer is almost full, so extra capacity is needed to prevent it from overflowing.

For each of the above-mentioned options, there are percentages of the fill level of the buffer attached which regulate what machines to operate and at what speed. For each scenario this is different, as each scenario aims to have a different type of output, and gives other machines more priority, especially at lower buffer levels. As soon as the buffer fill level enters a new range, the sub-scenario will change. Table 2.2 provides an overview of the buffer levels that are used per subscenario. Table 2.3 describes what the changes are per scenario, given a standard scenario where all machines are producing. This means that, for example in scenario 2, Machine three is switched off in subscenarios “fill buffer” and “Normal”, but will be switched on as soon as the buffer reaches the buffer level that belongs to subscenario “Empty”.

Scenario	1	2	3	4	5	6	7	8	9
Fill	0-5	0-20	0-20	0-20	0-20	0-20	0-5	0-20	0-5
Normal	5-10	20-30	20-30	20-30	20-30	20-30	5-20	20-30	5-15
Empty	10-40	40-50	40-50	40-50	40-50	40-50	20-30	40-50	40-50
Empty max	40-100	60-100	60-100	60-100	60-100	60-100	30-100	60-100	50-100

Table 2.2: Example of scenario (%)

As can be seen in Table 2.3 there are a few scenarios that are used during the collection period of the used dataset. The average speed and relative share of running each scenario is measured over data from one month in running the production line. Therefore, the average speed is highly influenced by the number of machine failures and its failure duration. This table is constructed from a dataset which provides data about which machine failed, how long the machine failed and which scenario and speed the production line ran. The period in which this data is collected is one month, November 2023. In this period, most of the time scenario 4 is used. Then Scenario 7 is sometimes used as an emergency scenario, where all machines are working. This is probably because in scenario 4, machine 3 should get preference, so machine 2 will be switched off when the buffer level is low. The scenario is chosen in advance, and per scenario, a decision is made when machines are switched off or back on again.

Scenario	Specific settings	Average speed	Share (%)
1	MC5 off	N.A.	0
2	MC3 on from Empty	N.A.	0
3	MC2 on from Empty	■	0.003
4	MC2 on from Normal	■	95.4
5	MC3 on from Empty, MC1 from Normal	N.A.	0
6	MC3 on from Normal	N.A.	0
7		■	4.2
8	MC1 on from Normal	■	0.14
9	MC2 on from Empty	N.A.	0
"Empty Buffer"		■	0.3

Table 2.3: Changes per scenario

2.3 Machine Standstills

This section describes the failure behaviour of the packaging machines, as well as the changeovers that will occur. For each machine, several statistics are calculated to get insight into the failure behaviour of each machine. For the buffer, no failures are taken into account, as this buffer is rarely failing. The buffer is further analyzed in Section 2.4.

The machines can be in different states, depending on their ability to produce products or not. There are five states which are distinguished regarding standstills. These are explained below.

- Running: The machine is operating and packaging products.
- Breakdown: There is a failure in the machine.
- Blocked: The machine cannot operate as intended, and is stopped.
- Operator stop: An operator has stopped the machine.
- Starved: The machine can produce, but has not taken any products.
- Changeover: The machine is changing its packaging material.

Within these states, there are three states where the machines are not operating. These are breakdown, blocked and operator stop. In the other two cases (running and starved), the machines are operating, whether they have a product that could be packaged or not. The reasons behind these states are not always applied in the same manner. For example, operator stops can have many different reasons. This means that we assume the status breakdown, blocked and operator stop mean that the machine is in failure, and thus is not able to package products. Also, because of the working of sub-scenarios, we distinguish another state. This is "stand-by". Stand-by means that the machine is not failed, but is not producing, in order to give priority to other machines. To summarize, the final states are explained below.

- Running: Machine is working.
- Stand-by: Machine is working, but on standby due to sub-scenario.
- Failed: Machine is not working and needs repair.
- Changeover: The machine is changing its packaging material.

Figure 2.3 shows the level of the buffer, compared to the number of machines available then. Each dot in the graph is the buffer level, given the number of machines at that point in time. A new data point is logged as soon as the number of available, working, machines is changing. At

that point in time, the buffer level is also logged. During the period of approximately 3.5 hours on the same day, a constant level of [REDACTED] cuts per minute was kept as speed. This figure shows a relation between the level of the buffer and the number of available machines. When there are fewer machines available, the buffer will likely be more filled than when there are more than 2 machines available. The case that the buffer was almost empty did not occur in the case of two machines or less, while this occurred multiple times when there were more than 2 machines available. Also, when there are more machines available, certainly with all machines available, at this speed level, the buffer will be less likely to completely be filled.

The buffer reaches its capacity in some cases. When there are either five or six machines available, the buffer hits 100% at some point in time. This means, that adjusting the speed might be necessary to prevent the buffer from overflowing, and the number of available machines is not the only influencing factor for the buffer level.

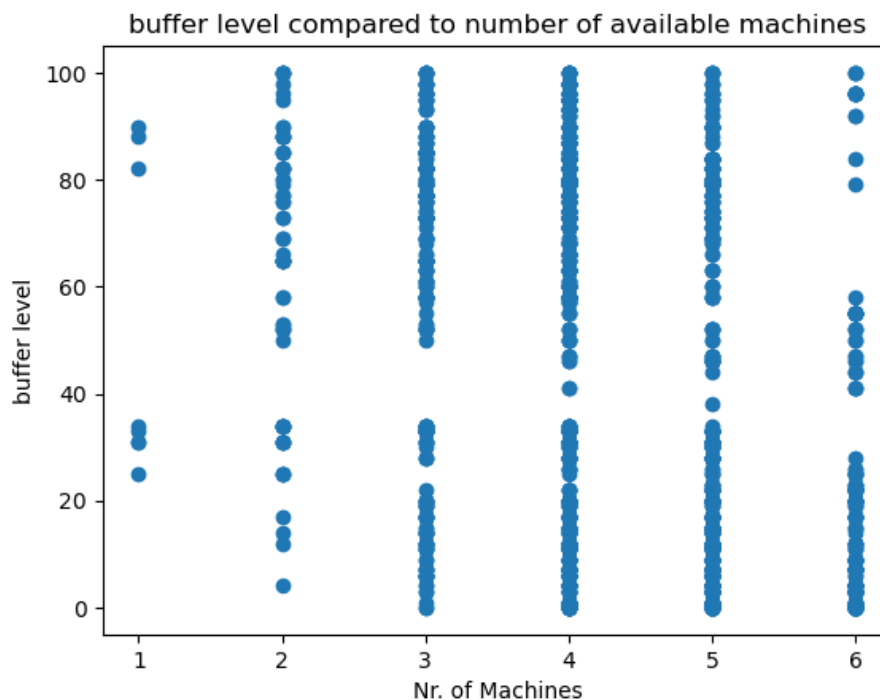


Figure 2.3: Buffer level compared to number of working machines

The number of machines available is thus changing often. However, it is not evenly distributed how many machines are working. Table 2.4 shows the relative frequency of how many machines are working at the same time, and therefore as well how many machines are failing at the same time. This table shows that most often, one machine is failing, however, almost as often, two machines are failing at the same time. For 16.9% of the time, there are no machines failing.

Nr. of machines	Percentage of time working
1 machine	0.66
2 machines	3.81
3 machines	13.59
4 machines	29.67
5 machines	35.28
6 machines	16.93

Table 2.4: Number of machines working

When a machine is failed, the duration before the machine is operating again is often unknown. Section 2.3.1 describes an analysis of the duration of machine failures. One last possibility why a machine can be at standstill is because of a changeover. A changeover occurs when a machine needs to change from packaging type. Section 2.3.2 describes the working of these changeovers.

2.3.1 Machine failure duration

When a machine is failed, the operators along the line try to fix this machine as soon as possible, to make it operating again. However, the time this takes is fluctuating among the different machines, and is not known before. Most of the failures occur at most three minutes, as Figure 2.4 shows.

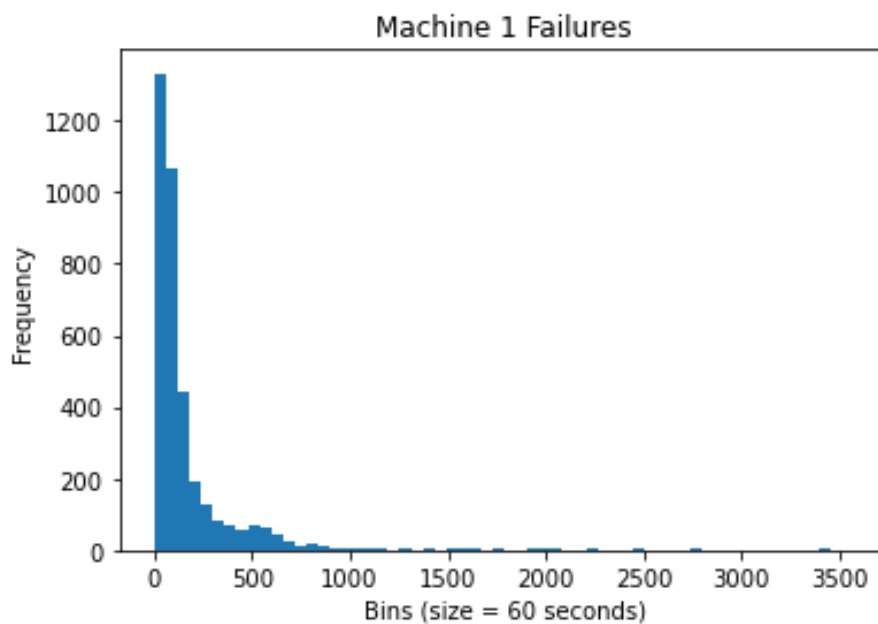


Figure 2.4: Machine failures

Graph 2.4 shows the number of failures, counted on frequency when grouped within bins. These bins are approximately 60 seconds wide. The graph in Figure 2.4 shows the failures for Machine 1. However, all machines show a likewise pattern. Appendix A show the graphs of the failed time for other machines.

The graph in Figure 2.4 shows that the range in which the failures can occur is large. The graph in this case only shows failures up to 1 hour. Longer failures are left out of scope, as we assume that when a failure takes longer than one hour, this is known upfront. The cause of many failures is not known, as many of the failures are just labeled as operator stop, without any further explanation of why this stop or failure occurred.

2.3.2 Changeovers

Another option of why machines are standing still is because the target level of production has been met. In this case, the machine should be changed to another package format. This has to be done manually by the operators, meaning that the machine should stand still for some period in time. However, due to uncertainty when machines take products of the line, especially for later products, it is not exactly known in advance when this changeover has to be made, making it more uncertain for decisions such as speed to be made. When speed is changed, changeovers in the near future should be considered, as these might have impact on the ability of the process to produce at that pace.

However, the decision of what the specific goal is per product package format, is determined in advance by the planning department, in the same way as the scenario that is selected. Therefore, the determination of these goals is out of scope, but the goals are considered themselves in the decision making.

2.4 Analysis of the buffer

This section describes the working of the buffer. The buffer is in place to prevent all items from being lost due to the unavailability of the packaging machines. Behind the buffer is one last machine, number 6, which processes all items that are in the buffer.

The buffer expands its size depending on the number of items in this buffer to prevent items from being longer than needed in the buffer. As soon as there are 15 rows in the buffer, the buffer starts processing rows of items towards the machine. This happens as soon as there is a new row of products entering the buffer. This means that a new row of bars is only proceeded to the machine as soon as a new row of bars arrive at the buffer. This procedure happens to prevent empty rows from occurring within the buffer, which eases the further process of operating the buffer.

The maximum buffer level is 385. In that case, all rows of the buffer are occupied and new products will be considered as waste, and will not be produced anymore. The logic of the buffer can be found in Figure 2.5.

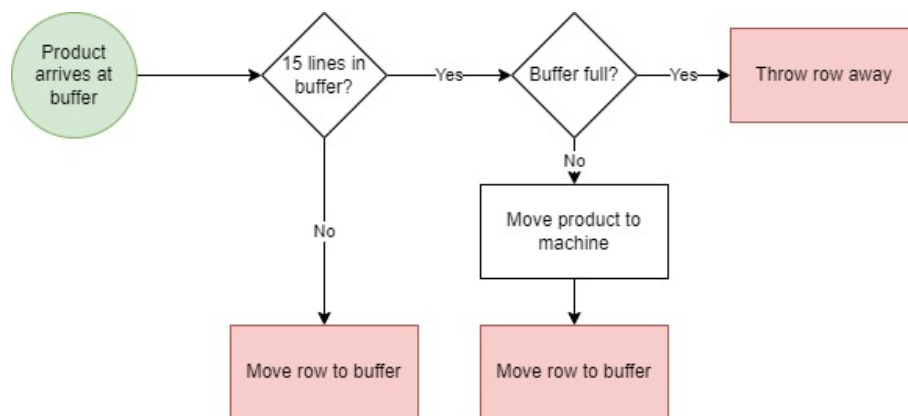


Figure 2.5: Flowchart of buffer

The buffer's failures are not taken into account, as it fails rarely. However, the use of the buffer is important to keep into account when deciding the input variables such as speed of the production line. If the buffer level is too high, measures should make sure that this buffer level is not increasing too much, while maintaining a high pace of production. Besides, the sub-scenarios as described in Section 2.2 take these buffer levels into account to decide about the desired input.

Figure 2.6a shows the effect of the speed on the buffer level. As can be seen in this graph is that as soon as the buffer reaches almost 100%, the speed is decreased. From this graph can be concluded that the speed is not the only factor that influences the buffer, but is a factor that can be used with certainty to decrease the buffer level. This graph also shows that the effect of a decrease in speed is not measured immediately, but will follow later. This is approximately 20 minutes before a change in speed will become visible in the buffer level.

Figure 2.6a shows the data from one day, between 04:00h and 11:00h, when the machine was already running, so without an initialization period. This graph shows that speed should be

chosen in such a way, that the buffer will not overflow, but also not be too low, because then a higher speed could have been performed. Also, before reaching 100% buffer capacity, the speed should have decreased already, to prevent the buffer from overflowing. However, the speed is not the only influential factor, as availability of machines is also important for the fill level of the buffer.

However, one point of discussion is that the speed is now decreasing with several steps, even when the buffer level is decreasing. This means that the question arises whether these decisions are optimal, or that the steps taken could increase, to get the buffer level down faster, or to increase the speed already in the end, to make sure that the machines produce more products.

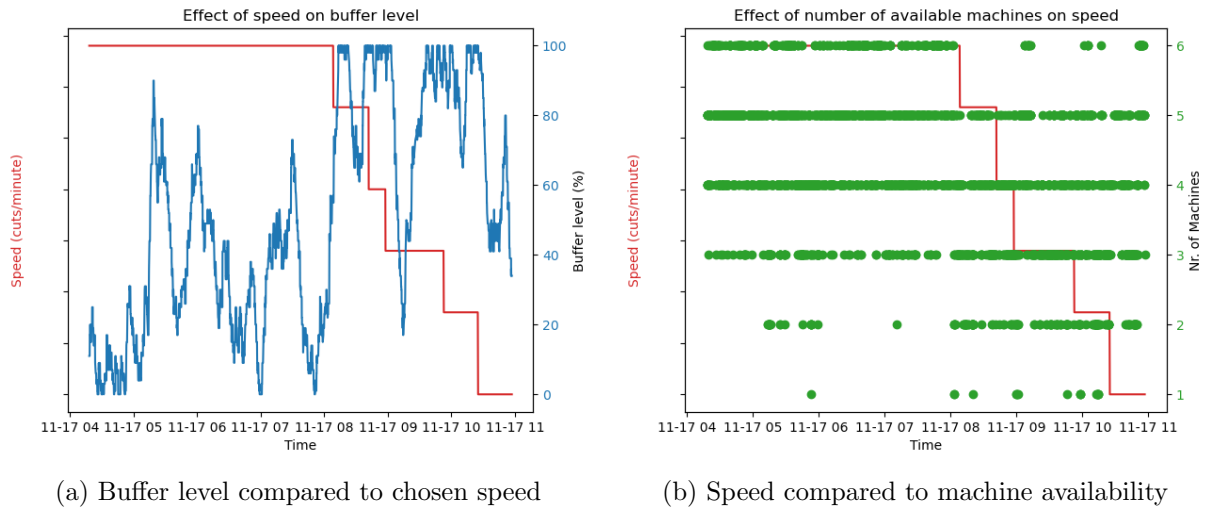


Figure 2.6: Speed, Machine availability and buffer level comparison

This means that the speed determination is dependent on the current state of the buffer, but also the availability of the machines. Figure 2.6b shows the decisions taken by the operators as soon as the availability of all machines goes down for a longer time. The speed immediately decreases, and, as concluded from Figure 2.6a, the buffer fills at this moment in time. This means, that the determination of the speed is based both on the availability of machines and the current buffer level. So, as soon as the availability decreases, also the speed should decrease.

The data in Figure 2.6 is in both figures the same, and represents therefore the same time steps and decisions made in speed. The buffer level grows to 100% as soon as only two machines are available for a longer time. The speed is kept at a decreasing rate because after the time the buffer hits 100% for the first time, it does not happen often that all machines are operating. The operators along the line have therefore decided to keep the speed low, and not increase it again.

2.5 Decision variables

This section provides an analysis of how and when the different types of input decisions are determined, together with what factors should be taken into account when making these decisions. These different types of input are the state of availability of the machines, the types of packaging, the speed of the different packaging machines and the number of items in the buffer. Also the progress towards goals is an important factor. All these factors have an impact on the output of the production process, both in terms of the number of items wasted and in terms of produced products.

Several factors determine the input characteristics of the production line. A specific speed should

be determined, together with the decision of which machines are functioning from which buffer level, and thus which machine or packaging material has priority. This depends on decisions made in advance and on the state of inventory and expected demand. However, this is also influenced by other variables as current speed and buffer level. The decisions on what scenario to run, so what item to prioritize is not included in this research. However, it is possible to take this scenario into account when making the decision on speed and processing sub-scenario. We explain the variables that have influence on the decision below.

- Speed. This determines the amount of products processed per time unit, and can be changed.
- Scenario. The scenario determines the priority and operating sub-scenario of the packaging machines. This has impact on a future speed.
- Availability of machines. Machines can be in error or maintenance, and therefore not working. This changes the behaviour of the processing line and the number of items to process.
- The number of items in the buffer. The fill rate of the buffer should be taken into account on how many products can be processed. Too many products in the buffer will mean a too high amount of waste.
- Already failed time. It is important to know the time that the machine is in failure when the operators will make a decision regarding speed. This time already failed has impact on the expected time that the machine will be failing, which means it has impact on the ability of the production process to keep up with a certain speed.
- Progress of goals. The progress of goals is the percentage of fulfilment in producing the goals per product type. This should be taken into account to make a more dedicated decision about future actions. For example, when one goal is almost reached, a changeover can be expected, meaning that a machine will be switched off during a certain period.

At the moment, the decisions which machines to operate are based on the scenarios as described in Section 2.2. However, employees working on this line are also able to make such a decision, for example when a machine fails while the buffer is increasing in percentage filled. Current decisions are both based on operators' intuition, combined with the current status of the buffer and machine availability.

The factors that the operators can have impact on are therefore the production line speed and the scenario. We therefore aim to provide a tool that is able to advice the optimal speed and buffer levels to increase production line output. The operators should not change the speed too often.

2.6 Conclusion

This chapter provided an overview of the current way of working of the packaging line and the way the buffer and machines are working. Also, an overview is given of the different scenarios that can occur within the production line.

This chapter gives insight into the many manual decisions that are made along the production line, which could be supported by the means of data. The determination of speed in comparison with the buffer level is something that could be done automatically, instead of using a operators intuition towards it. This provides a foundation for the literature review that is conducted in Chapter 3. Also, the different states of the machines are taken into account and are sampled to provide a clear structure between a machine that is unable to continue processing and a machine that can continue its processing.

Moreover, there are different variables to take into account while modelling the current situation. The availability of machines, number of items in the buffer, current speed and progress towards goals are factors that could play an important role in providing a calculated and data-based advice for the processing line.

For the speed, it is important to keep into account both the machine availability, as well as the buffer level. Both have impact on what is the optimal speed to choose. This should be considered, to decide on whether a speed change will be necessary in the future, or that the current speed is already optimal and should be kept.

The model which will be constructed needs to take care of these decision variables, and has to provide an accurate advice about this. Normally, the operators along the line take these decisions. The model needs to make these decisions based on historical data, and need to take into account the current states of machines, the current speed and current buffer level. It needs to decide whether this buffer level is a threat to eventual waste, and thus needs to decrease, or could fill up further, and thus a higher speed is possible.

Chapter 3

Literature

In this chapter, we discuss different models and methods regarding constructing a model to automate and optimize the real-time decisions that should be made regarding a production environment. Due to the frequently changing states of the production process, currently simulation, in combination with intuitive decisions, is used to get the best solution possible given an input state. This simulation model tries to copy the real characteristics of the production line as good as possible, as a so-called digital twin (Olcott and Mullen, 2020). This model uses real-world data, and provides advice for the real-life production line, which means that there is a double sided connection between the real production line and the simulation model.

First, in Section 3.1 we introduce the concepts of simulation optimization, ranking and selection, and multi armed bandits, in order to place this thesis within the literature. In Section 3.2 we discuss several methods that can be used to optimize a production line in real time. Also, we discuss control procedures for changing networks. Section 3.3 describes the concept of digital twins, which are used to represent real-life situations. This is needed to create a digital model which represents the real life production line. In Section 3.4 we discuss the principle of machine learning (ML), and how this can be combined with the digital twin. Within Machine Learning, many algorithms and methods exist that can be used. Section 3.5 describes some of these algorithms. This section aims to answer the following set of research questions and sub-questions.

2. *Resulting from a literature study, what solution methods exist to optimise the output of a production line?*

- What methods do exist to optimize the output of a production line given uncertainty?
- What are the possibilities to intervene in production speed to optimize output?
- How can Machine learning contribute to the optimization of the production process?

3.1 Background

In this section we introduce the concepts of Simulation Optimization, Ranking and Selection and Multi-Armed Bandits. These concepts are needed to place this research within literature. This research consists of a combination of these three concepts. We first discuss in Section 3.1.1 simulation optimization, before discussing the concept of Ranking and Selection in Section 3.1.2. Section 3.1.3 describes multi-armed bandits.

3.1.1 Simulation Optimization

Simulation Optimization approaches are used to improve efficiency of solving stochastic optimization problems (Xiao et al., 2024). The aim of simulation based optimization is to find a combination of the input factors that optimize an key performance indicator as output (M., 2014). When there is a large number of possible alternatives, it could simply not be possible to run and evaluate all the alternative configurations. There have been developed many procedures for searching through the space of possible input factors, for example metaheuristics as simulated annealing, random search procedures and gradient based procedures.

3.1.2 Ranking and Selection

In ranking and selection the aim is to select one of a certain number of alternative systems as being the best one, while controlling the probability that the selected system is actually the best system (M., 2014).

Ranking and Selection is used to identify the best design by using noisy simulation output (Wu et al., 2024). There are many sources in literature about how to select the best design using simulation estimates (Goodwin et al., 2021). An example of this is the probability of good selection (PGS) (Hong et al., 2015). Methods regarding offline simulation in order to take real time decisions are defined by Hong and Jiang, 2019, which uses context, in order to train a machine learning algorithm which predicts the output. They call this Offline Simulation Online Application. Shen et al., 2021 define this as Ranking and Selection with Covariates.

Ranking and selection with covariates, or contextual ranking and selection is further studied by Cakmak et al., 2021. Cakmak et al., 2024 defines ranking and selection as the problem where the best among a finite number of alternatives is selected, but the true performance is only observed through noisy evaluations. They define the context-dependent decision making in Ranking and selection methods when the best alternative can depend on this context. This is further analyzed by, among others Gao et al., 2019 and Shen et al., 2021. The context here is defined as covariates.

3.1.3 Multi-armed bandits

The multi-armed bandit problem is the online variant of Ranking and Selection. It is an issue where a trade-off should be made between exploring unknown solutions and exploiting best found solutions. It is stated as follows: there are K arms, which all have a fixed but unknown distribution of rewards. An agent plays an arm at each step. It receives a reward, which is independent of previous actions (Bouneffouf and Claeys, 2021).

Contextual multi-armed bandits, also sometimes referred to as bandits with side information or bandits with covariates, are a form of standard multi-armed bandits, which are studied by Lai and Robbins, 1985 and Auer et al., 1995, among others (Langford and Zhang, 2008). Contextual multi armed bandits have many applications and are more suitable than a regular bandit problem, because many settings do have context.

3.2 Production line optimization

Production lines are often dealing with many diverse problems, in different areas, different for every individual production line. Mathematical or computational approaches can be adopted to optimize the process of a production line as effectively and efficiently as possible. However, this might not always be possible due to the needed expertise, time and high computational cost (Kang et al., 2020).

Network optimization consists typically of two items: a network model and an optimization algorithm. The model predicts the performance and the optimization algorithm tries to generate configurations to meet this performance. One of the possibilities for network modelling is simulation. However, simulation requires high computational cost, making it often unable to operate at short time scales (Ferriol-Galmés et al., 2022). Short decision times are essential in the production line environment, where new decisions should be generated as soon as possible. Advances in Machine learning techniques have led to effective techniques for analyzing complex environments and solving problems in manufacturing (Kang et al., 2020). These models are trained on real-world data, which enables them to provide high accuracy predictions (Ferriol-Galmés et al., 2022).

By making use of new technologies it became possible to use virtual product and process planning (Kritzing et al., 2018). This resulted in large amounts of data, which after being processed, analysed and evaluated by simulation and optimization tools enabled planning in real-time (Boschert and Rosen, 2016). Advances in Machine learning, Deep Learning (DL) for example, can be used to train with real-world data and achieve accuracy by effectively modelling the complex network (Ferriol-Galmés et al., 2022).

Production planning and control (PPC) is the activity for creating and keeping a flow of production in manufacturing systems. It is about finding the optimal production quantity and processing sequence. PPC involves managing uncertainty in production systems via stabilizing production processes and proactive reacting on the needs of production. Dynamic performance optimization aims to monitor, evaluate and optimize KPIs continuously (Chiurco et al., 2023).

Production lines are dynamic and can encounter several disruptions or unforeseen events (Ghaleb et al., 2021). Dynamic real-time optimization (Daoutidis et al., 2018) becomes important in a frequently changing production network, where production control plays an important role. Decisions should be made based on current states and should provide the production line configurations. Optimization of control should be able to make decisions and balance productivity, sustainability, flexibility, and risk. This requires integration from various optimization and AI-based techniques. Simulation and regular planning and scheduling methods are less suitable for this real-time decision making (Tremblay et al., 2024) due to high computational cost. Therefore, more advanced methods are necessary to make real-time decisions at the production line.

Different data-driven models are available to simulate, optimize, and control processes (Petsagkourakis et al., 2020). Supervised learning models can predict behaviors and conduct process control. Reinforcement learning, with the use of a policy gradient method, can work directly with policies without the need for a model.

Artificial Intelligence (AI) and manufacturing systems can be combined to provide an accurate prediction measurement and evaluation of KPIs (Nazabadi et al., 2024). Machine Learning can be used to improve automation of manufacturing processes (Oriti et al., 2022). A challenge in training these models is that labeled data should be available. To overcome this problem, simulation models can be generated, which can generate data in an effective manner. These simulation models are used to create Digital Twin (DT), which can be used to improve predictions of future states, combined with simulation models and real-life data (Oriti et al., 2022).

3.3 Digital Twin

Today many companies have challenges with quick and accurate decision making. A solution to the time constraints often present in decision-making processes can be the use of Digital Twins (DTs) (Benfer et al., 2021). A DT is defined as “a virtual representation of real-world entities and processes, synchronized at a specified frequency and fidelity” by the Digital Twin Consortium (Olcott and Mullen, 2020). The virtual representation are the digital models and

their data, which provides the respective information. This definition of digital models is divided into two categories.

The first model consists of information that represents the states, entities and/or processes. The second is a computational model, a simulation model which consists of data, algorithms that get input and output for the representational models.

Digital Twins can be used to improve predictions of future states, support decision making, and optimize processes and operations (Oriti et al., 2022). There are four elements for digital twin development to meet the standards as set by the Digital Twin Consortium (Biller et al., 2022). These are data, domain, advanced analytics and outcomes. Data should be collected from all necessary states and types. The domain element combines the subject with modeling. Advanced analytics tools such as simulation can be used for performance prediction, data generation and analyses, by integrating this with machine learning and optimization. The outcome element means that the digital twin should be able to enable decision making, by providing more information, and automating the response at optimal settings.

Integrating DTs within information systems can offer the potential for problem-solving and decision-making support in manufacturing contexts (Chiurco et al., 2023). In this research, we collect data using a digital model, which takes its input statistics from the real life production line, and it is therefore important regarding the reliability of the collected data. Therefore it is important to see how a digital twin can play a role in the collection of the data.

3.4 Machine Learning

This section describes the concept of Machine Learning and its general algorithm division. With advances in computer technology, the ability occurred to process and store large amounts of data. Machine learning makes use of these large datasets, to process large amount of data, and take something valuable from it. This combined with Artificial Intelligence (AI), makes it possible for a system or environment to learn, to be intelligent (Alpaydin, 2010).

Machine Learning techniques are designed to gain knowledge from existing data (Alpaydin, 2010). A lot of different ML algorithms are available to implement in manufacturing processes (Wuest et al., 2016). One of the challenges is what ML technique or algorithm to select. The following approach can be followed (Wuest et al., 2016):

- Analyze how the available data is described, for example, whether it is labelled or not, to choose between supervised, unsupervised or reinforcement learning.
- Then, the applicability of the algorithms needs to be analyzed. For this, the structure, data type and amount of data that is available is important to take into account.
- Afterwards, the application of the algorithms to other problems has to be researched to identify if the chosen algorithm is suitable for the problem.

There are different types of machine learning algorithms, which can be used for different purposes (Kang et al., 2020). For the first step of the plan provided above, it is important to distinguish the different methods. These are as follows (Ayodele, 2010):

- Supervised learning: Deriving a function based on labels within data. An algorithm maps input to desired outputs. In a classification problem, a learner needs to map a function into several classes by looking at input-output examples of the function.
- Unsupervised learning: This type of machine learning does not require labeled data, but is used when relationships among data are not known. It does not receive an evaluation of the action that is performed (Monostori and Prohaszka, 1993).

- Semi-supervised learning: makes both use of labelled and unlabeled data.
- Reinforcement learning: observe the environment and perform a set of actions on this current environment. This produces a reward and the model is updated accordingly.

Machine learning has been used often to improve the quality of a production line, while the aspects of availability and especially performance are less researched (Kang et al., 2020). The following subsections describe supervised machine learning and reinforcement learning, as these are most applicable to the problem at hand.

3.4.1 Reinforcement learning

Reinforcement learning (RL) (Sutton and Barto, 2018) is a method that learns to maximize a numerical reward signal. The learner must decide which actions to take and must find out by himself which result to get from that. An agent takes an action $a_t \in A$, which is based on state $s_t \in S$, of the environment at time t (Overbeck et al., 2021). Here, the action space A describes the set of all possible actions to take, and S is the set of states. From an action, a reward r_t is received which leads to the next state $s_{t+1} \in S$.

Reinforcement learning has strength in learning actions based on data when there is no clear supervisor. An agent needs to learn by itself and from interaction with a system (Sutton and Barto, 2018). There is a trade-off between exploitation and exploration (Wuest et al., 2016). An agent has to exploit actions it learned to prefer. To identify those actions, the agent has to explore by trying new actions.

In the Multi-Armed Bandit problem, the decision maker selects an action from a given set (Balef and Maghsudi, 2023). After this selection, the player receives a reward. The player decides which action to take in a sequence of trials, to maximize its reward.

A subset of these problems are Contextual Bandit problems, where before the performance of an action, a context or state is observed. Based on this, the agent performs an action and receives its reward (Balef and Maghsudi, 2023).

Important for bandit problems is the selection of actions within a state. There are multiple strategies that can be followed. These strategies often balance exploration and exploitation. Besides the pure exploitation and pure exploration strategies, there is ϵ -greedy. With a small probability ϵ the agent selects a new action, that is not known to be best (Sutton and Barto, 2018). Other methods are upper confidence bound, where a confidence interval around the mean reward is constructed. Then, the action with the highest value of expected reward and confidence upper bound is selected as next action (Letard et al., 2024). Thompson sampling (Thompson, 1933), selects the next action to take using a beta distribution, which is based upon success and failures in previous iterations (Letard et al., 2024).

3.4.2 Supervised Machine Learning

Supervised Machine Learning is learning from examples that are provided by an external supervisor (Sutton and Barto, 2018). Within Supervised Machine Learning, there are again many possible algorithms to process the data. To select such an appropriate algorithm, it is important to analyse suitable ML algorithms on research problems that are similar to the current (Wuest et al., 2016).

A more mathematical representation is as follows (Hastie et al., 2009). There is an input variable \mathbf{X} , with output variable Y . Observed values will be referred to with lowercase i . Learning means that given the input vector X , a prediction should be made of the output Y . This prediction is denoted by \hat{Y} . Prediction rules should be established. For this, a set of training data is constructed, consisting of measurements $(x_i, y_i), i = 1, \dots, N$.

Supervised Machine Learning techniques can be divided into classification or regression algorithms. At classification tasks, the output is predicted within classes, it is a boolean output, yes or no answer (Alpaydin, 2010). In regression, the answer is a numeric value. The output is selected by a function. This function is not known, but a training set with examples of this function is. The aim here is to generate right output for a sample of input which is not included in training data.

For prediction in production process and control, common algorithms are the Artificial Neural Network (ANN), Support Vector Machine (SVM), Decision Trees (DT), K-nearest neighbour (KNN) or symbolic regression (Chiurco et al., 2023).

3.4.3 Model evaluation

Depending on the type of model, there are different methods to evaluate the performance of the ML algorithm (Chiurco et al., 2023). This section describes different methods and their uses.

- Mean Absolute Errors (MAE): This represents the mean of the absolute errors. If the value is lower, the accuracy is better.

$$MEA = \frac{\sum_{i=1}^n |y_i - x_i|}{n} \quad (3.1)$$

where y_i is the predicted value of observation i and x_i the actual value of observation i

- Mean Squared Error (MSE): MSE is a measurement that lowers as the error becomes smaller. The formula is:

$$MSE = \frac{\sum_{i=1}^n (y_i - x_i)^2}{n} \quad (3.2)$$

- Pearson's R^2 : A model is more accurate when its R^2 value approaches 1. The possible values are between 0 and 1.

3.5 Machine learning algorithms

In this section, we describe the various prediction algorithms that can be used on regression datasets. Due to the many applications of a deep neural network in multi-armed bandit problems, we choose this algorithm. Besides, we choose to take a form of decision trees, the Gradient Boosted Decision Tree algorithm. These algorithms are chosen because they have proven their ability to provide high accuracy in prediction (Jiménez-Gutiérrez et al., 2024). These methods will be explained in more detail below.

3.5.1 Gradient boosted decision trees

Gradient boosted decision tree (GBDT) is a machine learning tool which is widely used in classification, modeling and prediction (Zhang and Jung, 2021). The fact that this model is accurate, efficient and more easily to interpret makes it a suited model. It sums predictions of individual decision trees, which are the base learner of the method.

Gradient boosted decision trees produce robust procedures for regression problems (Friedman, 2001). It consists of separate individual decision trees. A decision tree is a model where the region or sample space is identified as a sequence of splits in a smaller number of steps (Alpaydin, 2010). A decision tree consists of internal nodes, places where decisions are made, and leafs at the end, where the prediction is made.

A decision node m consists of a function $f_m(x)$, which has discrete outcomes. When input is given, at each node, the function is evaluated and a branch is taken, depending on the outcome

of this function $f_m(x)$. This process, starting at the root, is continued up to the moment that a leaf node is reached. This consists of the output of the tree.

When a regression tree is constructed, the goodness of a split in a node is measured by the impurity measure (Alpaydin, 2010). This is defined as follows. For node m , χ_m consists of the subset of χ , which reaches node m . This is the set of all $x \in \chi$ satisfying the condition in the decision node on the path from the root until m . Then

$$b_m(x) = \begin{cases} 1 & \text{if } x \in \chi_m : x \text{ reaches node } m \\ 0 & \text{otherwise} \end{cases} \quad (3.3)$$

The goodness of a split is measured by the mean squared error from the estimated value. If g_m is this estimated value in node m , then $E_m = 1/N_m \sum_t (r^t - g_m)^2 b_m(x^t)$, where $N_m = |\chi_m| = \sum_t b_m(x^t)$.

If the error remaining in a node is acceptable, so $E_m < \theta_r$, the splitting stops and a decision (leaf node) is created. Otherwise, this process continues and the node is split further, to minimize the errors. A split is made with as goal that the difference in error, between not splitting and splitting in a branch, is maximized.

These individual trees can be used for boosting. Trees are sequentially added to the boosted model. For this, a loss function $L_y(f(x))$ determines how well a process is working at each step (Unpingco, 2019).

In gradient boosting, the loss function is defined as:

$$L(f) = \sum_{i=1}^N L(y_i, f(x_i)) \quad (3.4)$$

This can be used to optimize the following vector:

$$\hat{\mathbf{f}} = \arg \min_{\mathbf{f}} L(\mathbf{f})$$

Then, the sum of component vectors can be optimized for \mathbf{f} :

$$\mathbf{f}_M = \sum_{m=0}^M \mathbf{h}_m$$

With this result, a boosted decision tree ensemble is created, which is able to predict output on the given input.

3.5.2 Artificial Neural Networks

Neural Networks in general are models that are based on the central nervous system of a brain (Anzai, 1992). It can be represented as a distribution of neurons' states over the network.

A Neural Network is defined as follows (Anzai, 1992): U represents a set of processing units $U = \{u_1, \dots, u_n\}$. L is the set of links between two units. $L = \{(i, j) | u_i \in U, u_j \in U\} \subseteq U \times U$. All units are linked to other units by a link or from a node outside the network and send output to other links or the output side of the network. This can be defined, using output function g_i and state transition f_i , by output function as $o_j(t) = g_j(a_j(t))$ and state transition function as $a_j(t+1) = f_j(\sum_i w_{ji} o_i(t))$. Here w_{ji} represents the weight of link from unit i to j .

Artificial Neural Networks (ANN) are part of machine learning. They are comprised of several layers consisting of nodes. There is an input layer, an output layer, and in between some hidden layers. Each node is connected to another.

The process of decision-making within ANNs can be separated into four phases (Jäger, 2021):

1. Initialization
2. Experience
3. Training
4. Application

First, the initialization defines the features of the agents and their environment. In the second phase, random decisions are experienced upon which decisions should be made. In this phase, a pool of information is gathered. In the training phase, the network is trained upon a part of the dataset. The other part is used afterwards for validation. Finally, in the last phase, the ANN is used for the decision making.

Neural Networks can learn dependencies and exploit experiences already gained (Panzer et al., 2022). Learning is done by storing experiences and updating the strength of the connections. Inputs are processed and transformed into outputs that are immediately derived from recommendations, classifications or others.

Pairs of input data and their output are given to the neural network. By changing the weights of the links in the network, the preferred output of the model can come close to the actual output (Anzai, 1992).

3.6 Conclusion

From this literature review, we can conclude that there are many possibilities to create an online operational control procedure, that responds fast on frequently changing networks. A simulation model can be used as digital model that represents the real life production line. This digital model is constructed using data from the real production line. The use of Machine Learning methods can be beneficial to improve prediction accuracy, and use simulation data to be trained. Also, the time constraint, that solutions should be found as soon as possible, can be solved by implementing a machine-learning model. This will improve responsiveness to the problem and has the potential to create solutions faster than a regular digital twin, made in a simulation model.

Both the Neural Network and Gradient Boosted Decision Trees are applicable for pattern recognition and can handle large multidimensional datasets. This means that the algorithms can handle both multiple input as well as output variables. The ultimate decision about the performance of the models, besides the accuracy, is the run time. Neural networks have as advantage to be able to include difficult non-linear relationships in data.

This makes that both methods are well applicable in the problem presented in this thesis. Therefore, both the Neural Network as well as the Gradient boosted decision trees will be elaborated on with real data in the following chapters.

The problem central in this thesis is unconstrained in the training time, and can use a simulation model as long as possible and needed to optimize the performance. No assumptions are made about a linear relationship between input and reward, so more advanced machine learning algorithms will be needed. Another problem, present in literature, is the selection of samples to add to the training set. The entire possible dataset is too large for full observation, which means that selections need to be made in order to create a dataset using training time as efficient as

3.6. CONCLUSION

possible, resulting in a simulation optimization approach, where uncertain observations are used to optimize the functioning of a machine learning function.

Chapter 4

Solution method

In this chapter, we describe the solution method that is used to solve the core problem. First we describe multiple methods that can be used to solve the problem. Thereafter, we describe the chosen solution method. This chapter, together with Chapter 5 and Chapter 6 aims to answer the following set of research questions:

3. *How can a model be designed that fits best with the environment of the production line?*

- Which model does fit best with the available data?
- What input ranges should be considered within the model?
- How can we create a set of training data?

This Chapter describes the three different parts of the model: training, testing and using. Section 4.1 explains the procedure of training the machine learning regression function. Section 4.2 describes the approach of the testing and evaluation of this ML algorithm. In the end, we describe the idea of how to use the model in practice in Section 4.3.

4.1 Training

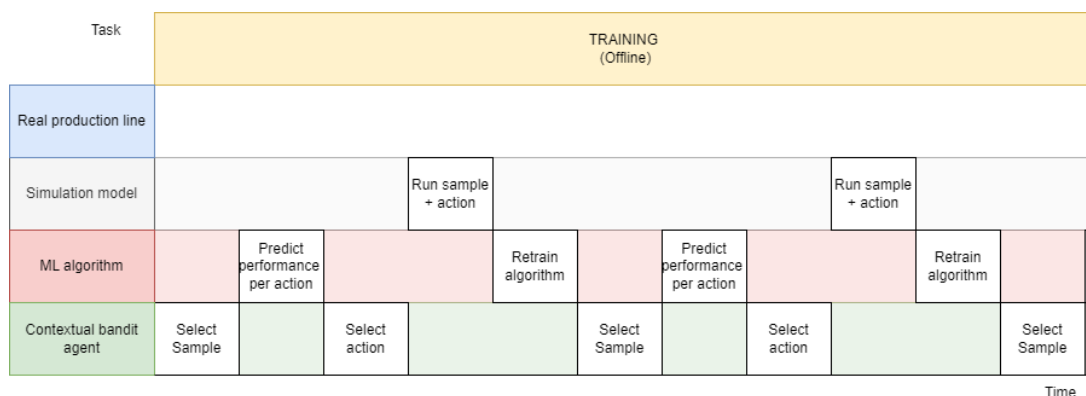


Figure 4.1: Training process

Figure 4.1 depicts the flow of the training process of the model. We train offline, because this speeds up the collection of data and this prevents making bad decisions with huge impact along the real production line. Taking an action that is not good is required for training, but in the real situation in the factory can cause large disruptions or waste production. This is not desired.

We start by initializing the machine learning regression function to make sure it can predict output on a given input. We initialize using a collection of 10 random samples. We run the simulation model on these samples, and train the machine learning algorithm on these input-output pairs.

We train the machine learning algorithm with contexts upon which this algorithm should predict the performance. A context consists of information about the state of the production line. Feedback, in the form of the real reward, is only given on the action that is selected. Therefore, the algorithm should be trained using methods that select these actions efficiently.

We train towards a policy. The algorithm selects according to this policy the best action given a context for training. This requires a balance between exploration and exploitation. During exploitation, sample selection should lead to choosing the action with the best expected reward, while in exploration should choose actions that potentially have a high expected reward. Exploration and exploitation therefore are both needed to improve the prediction function, and to find other actions that might improve the performance, while also the best known action is sometimes chosen to improve the policy.

The simulation model is finally used to run this sample with the chosen action. We observe what the simulation model gives as output, and use this information to retrain the machine learning model. We continue this process up to the moment that a stopping criterion is met, for example we used all training time or have reached a determined number of iterations.

To prevent the retraining of the Machine Learning algorithm to take too long, we limit this retraining as much as possible. This means that the Machine Learning algorithm is updated with the newest information once every ten iterations. This reduces the time needed for retraining, which means that more samples can be evaluated during training.

The following subsections describe the actions that can be selected and the way of selecting these actions. Section 4.1.1 describes the algorithm for selecting which action to simulate. The actions that can be taken is explained in Section 4.1.2.

4.1.1 Contextual bandit

We describe the mathematical model of the contextual bandit. Define \mathbf{X} as the set of input data, represented as a matrix, consisting of different samples of vector \mathbf{x}_i where $i = 1, \dots, n$. Each individual input variable within a sample is the variable x_{ij} , where $j = 1, \dots, m$. So, there are in total n samples in the dataset and m variables which represent the production line state within a sample. The target output is defined by Y_i , corresponding to input sample \mathbf{x}_i .

We also define the set of all decisions that the agent can take, where decisions range from $k = 1, \dots, K$. We define a function $f(x_{i|k}, Y_i)$ that predicts the output to the given input. This means, the function should predict the output on sample \mathbf{x}_i when decision k is taken. The objective then can be defined as taking the optimal decision k^* under a given context \mathbf{x}_i , which is defined in the following equation:

$$k^*(Y_i) = \arg \max_{k \in \{1, \dots, K\}} f(\mathbf{x}_{i|k}, Y_i) \quad (4.1)$$

which is the optimal action k^* under context vector $\mathbf{x}_{i|k}$. This is the optimal action, given a specific sample as input. In this equation, k are the possible values of speed that the system can take. The context \mathbf{x}_i is a sample observed by the real line. By training the function $f(\mathbf{x}_{i|k}, Y_i)$ the agent should be able to provide correct decisions on the speed of the production line.

In the model described above, not all contexts can be observed by the decision maker during training. Based upon historical information, the decision maker should be able to take decisions which are as good as possible. Contextual multi-armed bandits provide a method of training that makes the machine learning algorithm able to provide decisions regarding these unseen contexts. The method of selecting the action given the objective above is modelled as follows.

A context \mathbf{x} arrives at the moment the decision should be made. An agent, the entity that should provide advice about the next step to take, decides to take action $k \in \{k_1, \dots, k_K\}$, where K is the number of actions the agent can take. With action k in context \mathbf{x} , the agent will receive a reward $r(k, \mathbf{x}; \Theta)$, where Θ corresponds to the randomness that is represented in the achieved reward. The objective of the agent is to create a policy π that maximizes reward $r(A, C; \Theta)$ for taking action k in context \mathbf{x} while observing some randomness Θ .

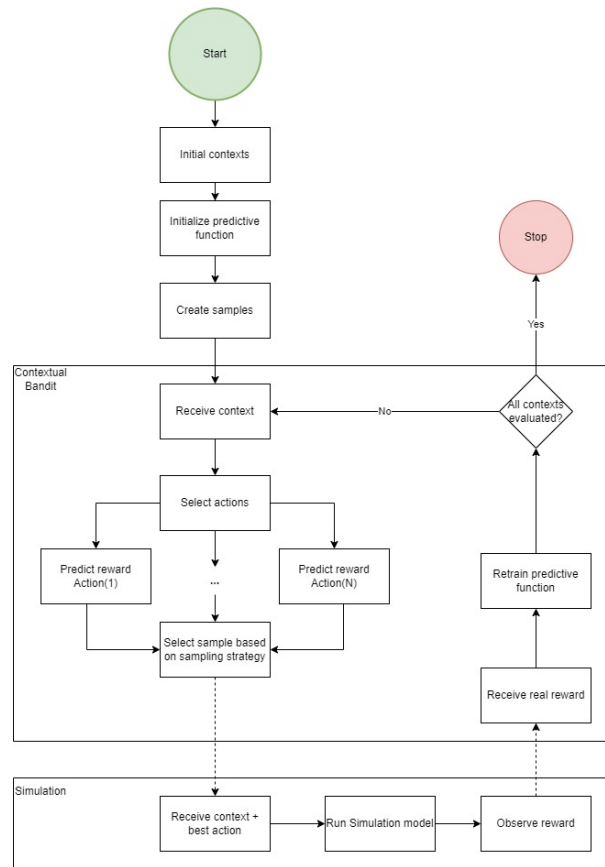


Figure 4.2: Solution approach training

Figure 4.2 shows how the contextual bandit and simulation model work together in more detail. As mentioned before, it is not possible to train the algorithm on all samples, so a selection needs to be made. In order to review an as large as possible part of the complete sample space, we provide random selected input samples. On these samples, the machine learning algorithm predicts the reward on each action. The action that is selected is provided to the simulation model, which presents the output back to the Machine Learning algorithm. With this iteration, we keep training.

Per sample, the agent should take an action. Which action to take, is determined by a sampling strategy. This strategy takes care of the exploration/exploitation trade-off. Examples of these strategies are ϵ -greedy, Upper confidence bound or Thompson Sampling. The agent will only receive the real reward of the combination of the sample and the action selected to evaluate.

We provide the algorithm for the selection of actions in Algorithm 1. This algorithm has as goal

to only add samples to the set of training data that provides valuable information to the agent. It not only receives feedback on actions that are expected to give a high return, but also take actions that are possible to receive a high reward, and therefore can influence the performance of the agents predictions. This means that only the most uncertain samples and actions are added to the training set, or actions and samples that are expected to perform good. Which action and sample is added to the training set is determined by the strategy that balances the exploration/exploitation dilemma.

Algorithm 1 Contextual Bandit

```

Initialize model, determine batch size and number of actions "NumActions"
while KeepRunning = True do
  for  $i := 1$  to  $BatchSize$  do
    observe context  $X_i$ 
    for  $j := 1$  to  $NumActions$  do
      if Algorithm = "Neural Network" then
        for  $l := 1$  to 30 do
           $y_l := f(x_i, k_j; \theta)$ 
        end for
         $\mu_{i,j} := \sum_{l=1}^{30} y_l$ 
         $\sigma_{i,j} := \frac{\sum_{l=1}^{30} (y_l - \mu_{i,j})}{30-1}$ 
      else if Algorithm = "GBDT" then
        Calculate prediction interval
        Calculate mean prediction  $\mu_{i,j}$ 
      end if
    end for
    Select action based on sampling strategy ▷  $\epsilon$ -greedy or softmax
    Add sample with selected action to training set
  end for
  Evaluate sample batch using simulation model
  Retrain prediction function
end while
  
```

Algorithm 1 shows the general idea of how the agent selects a sample and is updated. First, some initial samples are provided to the prediction function, with their real rewards. Based upon these samples, the agent is initialized. Then, a batch size is determined. This batch size is the number of samples that are selected to add to the training set, before the model is retrained. This batch size is used to speed up the process of updating a prediction function. For each selected sample and action, the reward is predicted, and the uncertainty of the model on this prediction is determined. Based upon a selected strategy, for example ϵ -greedy, samples are added to the set with training data.

It is important to take uncertainty within predictions into account, and not just select the action with the highest expected reward. Actions that are uncertain might produce high reward, and are therefore valuable to evaluate as well. For this, a balance should exist between exploration and exploitation. There are several strategies to balance these two. Common strategies are ϵ -greedy, where with a probability of $1 - \epsilon$ the best expected predicted action is taken, and with a probability of ϵ another sample is selected. Another is Thompson Sampling, which provides a way to select, based on randomness, a promising action that not necessarily has the highest expected reward. Another way to overcome randomness in observations and to incorporate the ranking and selection method is using the Boltzmann equation (Powell and Frazier, 2008). For each action, a probability, $P(a_i|x)$ is calculated that this action could be picked, based upon the required exploration/exploitation trade-off, and the promising expected value. We use the

following formula, where \hat{y} is the expected value, and ρ is a tuneable parameter which following a cooling scheme.

$$P(a_i|x) = \frac{\exp(\hat{y} \cdot \rho)}{\sum_{k=1}^j \exp(\hat{y} \cdot \rho)} \quad (4.2)$$

When ρ is small and reaches 0, the probabilities tend towards the highest rewards, increasing probability that those will be selected. In case when ρ increases and becomes large, the probabilities become more uniform, leading to larger exploration.

The strategy used for this is to explore in the beginning, to let the Machine Learning algorithm get a good image of the sample space and interactions of the speed actions and rewards. When training time has elapsed, the probability of exploitation will increase, to provide more information about the real rewards of the best predicted actions to the machine learning algorithm.

4.1.2 Actions

We use the contextual bandit algorithm to select actions that can be taken on the current state of the production line. We present the contextual bandit with a random selected sample. On each sample, we use the machine learning algorithm to predict the output of all different actions that can be taken.

Then, based on a strategy that we choose to handle the exploration-exploitation trade-off, the contextual bandit selects one action. This action is either the best action, and will provide the machine learning algorithm with more information about the performance of this action, or is an action that is more unknown, and will therefore provide the machine learning algorithm with a new opportunity about the reward, and may improve the policy of selecting actions.

The actions that the agent can take are the speed of the production line, together with the buffer level ranges when machines should be switched on and off. The operators along the line are able to change this speed and buffer level ranges. The speed can range between \blacksquare cuts per minute and \blacksquare cuts per minute. To reduce the total amount of options, only step size 2 in speed is considered. The buffer level ranges can lay between 1 and 100%. However, the buffer level when machines should be switched on should be higher than the buffer level when machines should be switched off. This action type is to make sure that the buffer does not overflow, but increases the flexibility to actually produce the needed products. Table 4.1 provides an overview of the actions that can be taken.

Action	speed	Action	machines ON/OFF
1	\blacksquare	Buffer level	0-100
2	\blacksquare		
...	...		
11	\blacksquare		
12	\blacksquare		
13	\blacksquare		

Table 4.1: Actions to take

The actions are categorized into two groups. The first action the agent should take is about the line speed. This ranges between \blacksquare and \blacksquare . The second action the agent can take consists of the buffer level when machines can be switched on and off. The buffer levels replace the “scenarios” which determine when machines should be switched on and off. Instead of a

predetermined schedule, the agent should decide at which occupancy level of the buffer it is possible to switch machines off, or back on again. For this, we choose 1 level of buffer fill rate, which provides the switch. In order to prevent from switching on and back off again, we make this a region of 10 %, which will be the trigger levels.

4.2 Testing

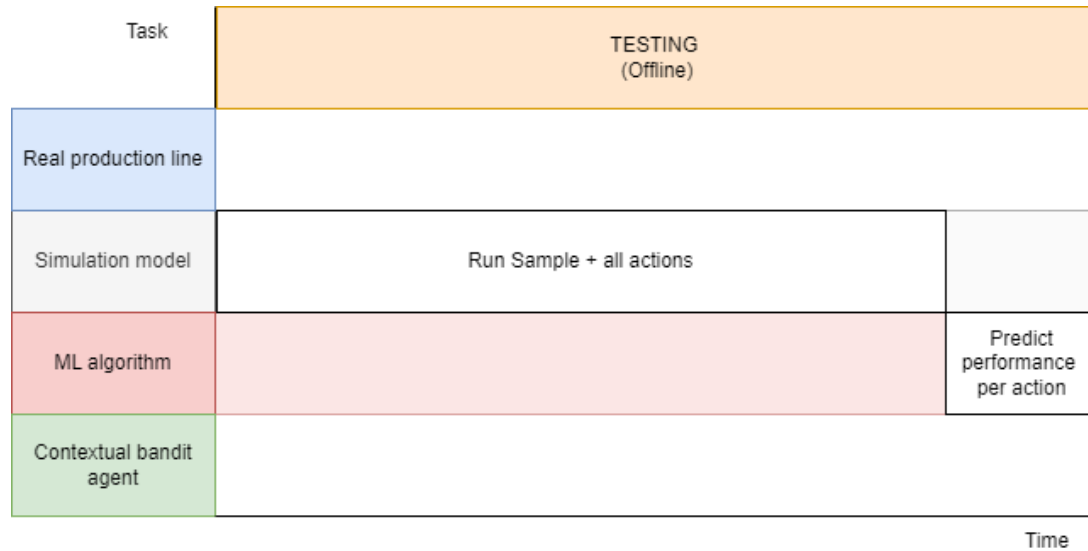


Figure 4.3: Testing process

As soon as the training phase is over, we go to the testing phase. Figure 4.3 shows the elements of this process. Testing is still done in the offline phase, because the real production line is not available. In testing, we use again the simulation model to run a sample, which is evaluated over all actions. However, now we use the simulation model to provide the best advice on a production line state. The simulation model calculates for all actions the production output and waste, and determines the best action to take.

We then use our machine learning algorithm to predict the performance of each speed and evaluate the results. We use two types of validation. The aim is to create a tool that can predict as accurately as possible the required speed. We evaluate the performance of the machine learning algorithm on different samples, and measure the amount of times the prediction is done correctly.

Advice can be generated upon request of the operators, or each pre-determined time period. In order to make sure the production line is not running on the wrong speed, the time intervals between two consecutive advises cannot be too long. However, the time after which the speed change is observed is 20 minutes. So, making more speed changes in between is not desired. We will provide experiments that determine the optimal time interval that is required to determine the optimal speed.

4.3 Using

When training and testing is over, and the model works as desired, we are able to implement the model at the real production line. Figure 4.4 shows the process over time of this process. The real production line shows the real state of the line. With this state, we use our machine learning model to predict the performance of each action to take. We select the action that is

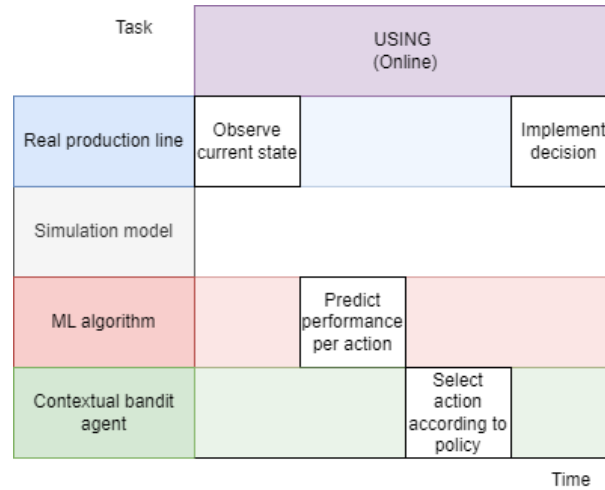


Figure 4.4: Advising process

predicted to be best. In this phase, the model is fully trained, so there is no need anymore for exploration of new solutions. We therefore select only the action that is predicted to receive the highest production reward.

4.4 Conclusion

In this chapter we described alternative methods to solve the core problem. We chose one solution method, which is based on the idea of contextual multi-armed bandits. This is a procedure, which uses rewards to determine the next action to take. This sampling strategy is based on a trade-off between exploration and exploitation. We use the simulation model to evaluate the selected context and action. Because of the large action space, not every context can be evaluated by the decision-making agent. This means that a smart sampling strategy should be used, where samples are added to the training set based on a exploration-exploitation trade-off.

Chapter 5

Simulation modelling

In this chapter we describe the conceptual simulation model. As mentioned in Chapter 4 the simulation model is used to evaluate given scenarios for an agent that provides advice to the operators along the production line. A technical description of the flow of products through the simulation model and procedure of modelling can be found in Appendix B. In this chapter we answer the following set of research questions:

4. *How can a simulation model be designed that evaluates the performance of the production line?*

- What assumptions are made during simulation modeling?
- What type of simulation model is needed?
- What factors should be considered in the simulation model?

The simulation model consists of a conveyor belt. We show the visual representation of this simulation model in Figure 5.1. Products enter this conveyor belt from the process department, and are packed by the machines. Products are transported across the conveyor belt, starting from processing. Machines take these products, and pack them. If that is not the case, then products end up in the buffer. If this buffer is also filled, the products are thrown away as waste.

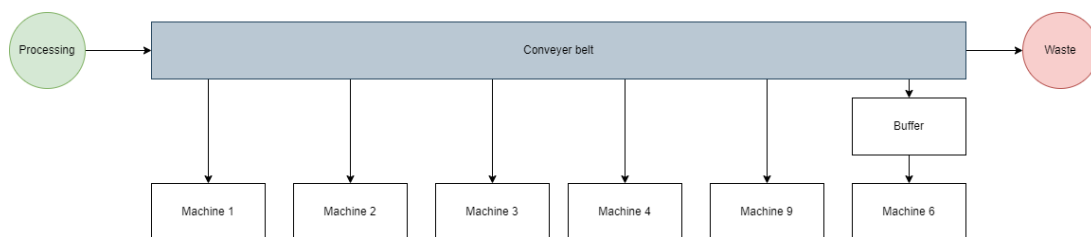


Figure 5.1: Simulation model lay-out

The simulation model is a model with transient behaviour. We are interested in the performance after an initial starting state. We also treat the simulation as terminating, because we are only interested in the behaviour of the simulation model after the start state up to a certain moment in the future.

The following subsections describe the different elements of the simulation model. We start with the modelling assumptions in Section 5.1. In Section 5.2 we describe the input variables which are used to create a sample, in Section 5.3 we describe how the model is initialized with

these input variables. In Section 5.4 we describe the objective function which is calculated per simulation run and given as output of the simulation model.

5.1 Modelling assumptions

In this section we describe the assumptions made during modelling. The assumptions are listed below. Because the model is designed as a digital twin of the real-life production line, not many assumptions are taken into account.

- There is always input of products. There is a constant flow of input from the processing department.
- We expect that the buffer does never fail.
- Waste is only considered when the buffer is full, other types of waste along the line is not taken into account.
- Only the first wrapping action is taken into account, as this is the process that keeps the machine occupied.
- Machines fail up to 30 minutes. If a machine is already failed for 30 minutes or longer, the machine will be switched off for the entire simulation run.

5.2 Input

The simulation model uses several variables and parameters as input. In the following sections, we describe this input data, and the way it is used in the simulation model. Table 5.1 shows the input variables, their ranges and datatype. It also shows whether the variable or parameter is changeable during the process or fixed in advance.

Input Variable	Range	Datatype	Comment
Speed	■■■■■	Cuts/min	Fixed by operator
Current buffer level	0-100	Percentage	Measured during process
Current state machines	Working, failed	Categorical	Measured during process
Current Scenario	1-9	Integer	Fixed by operator
Current Item	ProductItem	PackFormat	Fixed in advance
Time already failed	0-3600	Seconds	Fixed in advance
Progress to goal	0-100	Percentage	Measured during process
Next item	ProductItem	Packformat	Fixed in advance

Table 5.1: Input variables Model

We provide input to the simulation model in terms of variables. We refer to this set of input as a sample or context. A sample serves as the starting state of the simulation model. It is the state of the production line which is seen at a specific moment in time, when a decision should be made. The simulation model only observes the current situation, and evaluates what will happen in terms of production and waste. This output is in terms of an objective function, which takes into account the waste and production per product type that is produced in the hour following the begin state. This means that for each starting situation, a sample, that is given as input in the simulation model, an output is created in terms of the objective value.

The input variables in Table 5.1 describe the starting state of the simulation model. The current speed is determined by the operators and can be changed when the production line is operating. The current buffer level is observed during process, and cannot be changed directly. The same

holds for the state of the machines. This variable represents whether the machines are working, failed or in standby.

Whether machines are in standby is determined by the scenario. This scenario is a specific processing schedule, that determines what machines can be switched off temporarily, and at what buffer level this should be done. It also determines at what buffer levels these machines should be switched on again, to prevent the buffer from overflowing. Scenarios are predetermined by the planning department.

Each machine packs a current item. Items are distinguished by the packaging material. Each item has a specific goal for the number of items to produce, which is according to the pre-determined production schedule. When this goal is reached, there occurs a changeover of packaging material within a machine. Then the machine will start producing the next item on the production schedule. When the packaging materials are different, the changeover takes longer. This means that the machine at which the changeover takes place is switched off for a longer period. If packaging materials are not different, changeovers take a shorter time.

If the machine is failed, the variable “Time already failed” consists of the time that the machine has failed already. This is important for the prediction of how long the machine will remain failed in the simulation run. Otherwise this is 0. There are, as we described in Section 2.3, machine failures at random moments.

For the failure times of machines we use the mean time between failures (MTBF) and assume this at 11 minutes and 31 seconds for all machines. It follows an exponential distribution. In the starting state, we provide a failure time, which is the time that the machine is failed up to the moment that the starting state is measured. Historical data is used to determine the time that the machine will remain failing. This data consists of failure times up to 30 minutes. When a machine is failing 30 minutes or longer, we assume that the machine will be failed for the complete duration of the simulation run. Most failure times are, as we explained in Chapter 2 shorter than 30 minutes. We consider the failure behaviour of all machines the same.

We calculate the machines failure time from this list of historic failure times. First, we sort the list and remove all times shorter than the time the machine is failing. From the remaining failure times, one is randomly selected for the machine. The difference between this selected failure time and the time the machine is already failed is the time of the remainder of the failure.

5.3 Production process

The output of the simulation model will be evaluated based on the starting state. The simulation model predicts what will happen during production. The products flow over the production line, and are packed by the packaging machines. As in the real situation, the products are taken from the line if the machine is empty and working. Otherwise, products will be left on the production line.

In Figure 5.2 we visualize the logic of when a product is picked by a machine, and when this is kept on the conveyor belt. This flowchart starts as soon as the product enters the conveyor belt and reaches the exit to a machine. It determines whether a machine should be packed by that machine, or needs to stay on the conveyor belt. Each time a product transports along a machine entry, this flowchart is followed to see whether the machine can pack these products.

If products on the conveyor belt come across a machine, and this machine is available, the machine picks the products. Only if the machine is the last machine, the products are entering the buffer. If the machines are not available, the products remain on the conveyor belt. If the buffer is also not available, the products end up at the waste bin.

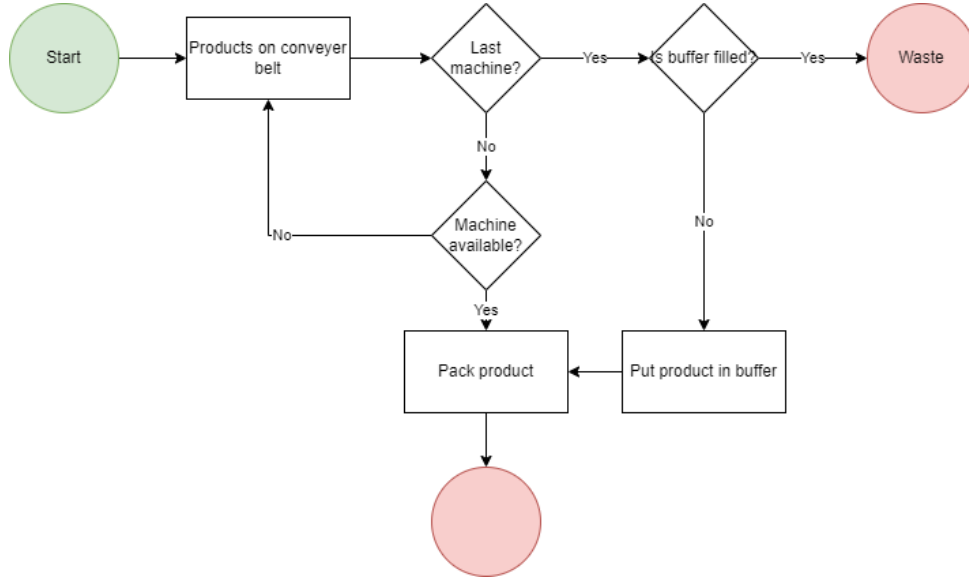


Figure 5.2: Machine logic

Depending on the buffer level, machines can be switched off. If the buffer level is sufficiently low, the machine with the lowest priority can be set to standby. This machine will start working again if the buffer level is reaching an upper bound. These values depend on the scenario, which is an input variable.

The subscenarios, which determine at which buffer levels the machines are switched off and on are given in Appendix C, where the tables provide these buffer levels and the machine statuses that correspond to these buffer levels. Scenario 7 provides the situation where each machine keeps running, which means that no changes are made depending on the buffer levels.

However, when these scenarios are omitted, the simulation model determines itself which machine to set standby. The simulation model calculates a priority for each machine. We use the following formulas to calculate the priority per machine. The time to target is calculated per machine, which determines which machine will take the longest in the coming period for finishing its production schedule. Then, based on this time to target, the product with the highest time to target given the expected end time of total production is given the highest priority. When a machine is done producing very fast, then this item gets a low priority.

$$TimeToTarget_i = \frac{TonsToGo_i}{\frac{SpeedMachine_i}{SOC_i}} \forall i \in Machines \quad (5.1)$$

where SOC is the standard operating capacity, which is assumed to be the availability of the machines in this situation, which is assumed to be 85% for each machine, meaning that 85% of the time, the machines are working. Failure duration is determined randomly, based upon historic data of failures.

$$Priority_j = \frac{TimeToTarget_{i \in j}}{ExpectedEndTimeOfProduction} \forall i \in Machines \quad (5.2)$$

where the expected end time of production is determined by the sum of all times that machines will be working on their current product type.

The machine with the highest priority will be left running, where the item with the lowest priority is evaluated to be switched off. This means that each time the buffer reaches a certain

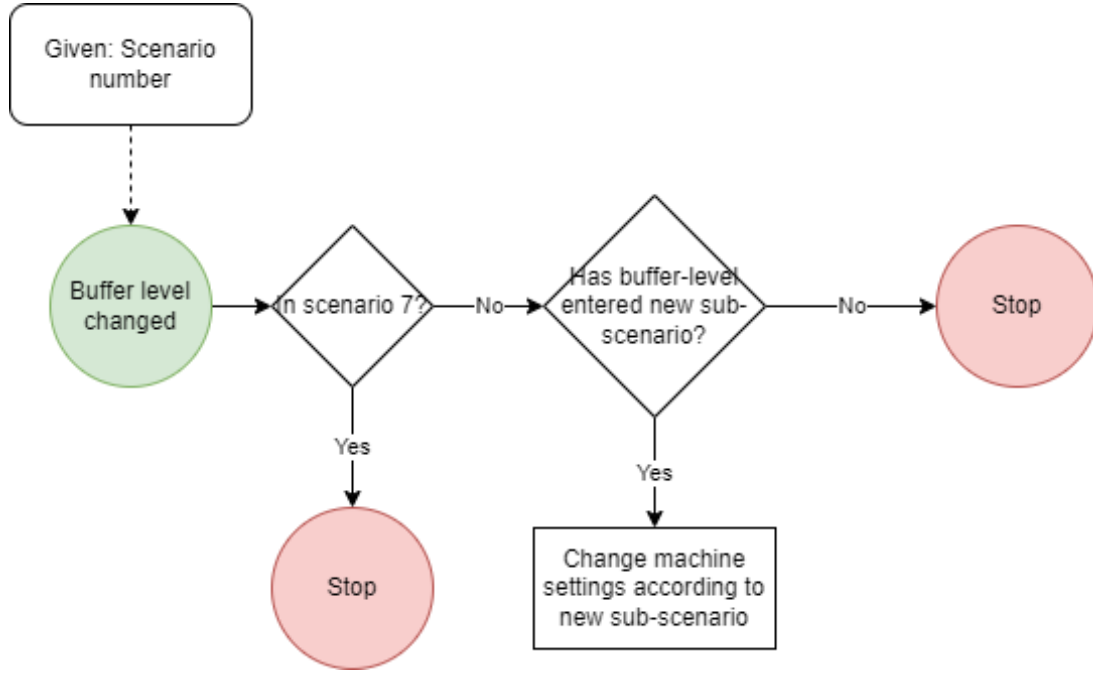


Figure 5.3: Flowchart determining new subscenario

lower bound, this machine will be stopped. Each time the buffer exceeds a maximum value, the machine starts operating again, to prevent the value of waste to increase.

5.4 Objective

After an experiment is performed, the output of the simulation consists of the number of products, measured in tons, that are produced. Also, the items that are wasted are counted. This is combined into a single value, depending on how the company values waste and production. This might change per production batch, due to for example demand expectations, current inventory levels and production deadline.

Per experiment, an objective value is calculated. Formula 5.3 provides the mathematical representation of the objective function. This formula consists of the weighted amount of produced products, minus the weighted amount of waste that is produced.

The product types (multipack and showcase) are separated in this objective function. These two product types are produced on different machines, which means that the itemcodes that belong to these product types are grouped in two groups. Machines 1, 3 and 9 produce showcases, while machine 2, 4 and 6 produce multipacks. Because of the same nature of these product types, we select them as being evenly important in production for demand. Therefore, we only consider these two groups of product formats in the calculation of the objective function.

$$Z = \arg \max_{k \in \{K\}} \sum_{i=1}^2 (x_i \cdot P_i) - TW \cdot C_{TW} \quad (5.3)$$

where

- K = set of possible speeds
- x_i = production in tons of product type i .

- P_i = weight of product type i .
- TW = Total waste of production, in tons.
- C_{TW} = Penalty cost of producing waste, per ton waste.
- $i = \{0, 1\}$ where $i = 0$ if multipack, $i = 1$ otherwise.

The objective balances waste with production. As long as the benefit of producing an extra item is larger than the cost of wasting another item, speeding up the production process will be beneficial. The objective calculates the maximum value of the output of the simulation model, given the input parameters. The output parameters that are changed in order to evaluate the best performance are the future speed and the fact whether machines should standstill or be producing.

The simulation model has as aim to evaluate the objective value given the input variables as described before. The main collected variable is the production output and waste, which the model observes given an input situation. Together with all the input data, this is collected into one dataset. For each sample that is given as input, the simulation model provides an estimate for the output. As mentioned before, the output is not deterministic, because of the stochastic behaviour of machine failures.

Due to randomness in the simulation model, where machines behave in a random manner regarding failures and failure duration, the output given by the simulation model is stochastic. The simulation model uses a random number stream, which means that each time the simulation is started using a different random number stream, the results of the run are different. However, when this random number stream stays the same, the random events also stays the same. The objective value is therefore depending on the randomness that is incorporated by the simulation model. This means that the output given by the simulation model after just one run is not necessarily what will be seen in the real situation. Therefore it is important to take some uncertainty with these values into account. It cannot be set as optimal or deterministic value, because each time a context is evaluated, some other types of failures can occur, which changes the output of the simulation model. To overcome this randomness, we can repeat the same experiment multiple times. This can provide a confidence interval around the mean, which enables us to provide the final solution.

5.5 Conclusion

This chapter described the working of the simulation model. This simulation model creates data, that should represent the real life production line as good as possible. For this, we described the input that will be given to the simulation model, which variables are important to take into account when deciding on the best speed and machine functions for the next period. We described an objective function which calculates a balance of waste and production, and takes into account which type of products are produced. It takes weights, which influences the relative importance of waste and production, and the product types among each other.

The input is given to the simulation model. This input represent the status of the line, from which it will start the simulation run. We defined an objective value that calculates a value based on the amount of production and waste, based upon weight and importance of production.

Chapter 6

Machine learning algorithm

This chapter describes the design of the machine learning algorithm, using data that we collected via a discrete event simulation model. This chapter provides an overview of the solution method that is used to eventually provide advice to the operators along the production line to improve the production line performance. This chapter answers the following set of research questions:

5. *How can production output and waste be predicted?*

- How can functions be trained to predict output?
- How can decisions be made from the prediction?

In this chapter, we first describe the data preparation techniques for the prediction functions in Section 6.2. Then, we describe the methods of cross-validation and preventing over-fitting in Section 6.3. Next, we describe the two different machine learning methods in Section 6.4.

6.1 Data analysis

In this section, we discuss the data that is available and run a tool that determines which machine learning algorithms are most applicable for this type of data. Based upon a selection of random data samples, where input is randomly selected, and evaluated by a discrete event simulation model on the number of production and waste, RapidMiner, a data analysis tool, (“Altair RapidMiner”, n.d.) can determine which models fits best with this data. This program models different algorithms automatically, and optimizes the models. Table 6.1 shows the performance of several chosen machine learning methods on the mean squared error and their running time.

Model	Root MSE	training time
Generalized linear model	0.022	81 ms
Deep learning	0.021	458 ms
Decision tree	0.023	17 ms
Random forest	0.027	19 ms
Gradient boosted trees	0.017	261 ms
Support vector machine	0.024	397 s

Table 6.1: Performance of algorithms

From this table we can conclude that the Gradient boosted trees perform best, but is also a model that requires longer time before optimizing and training. This time is however not a constraint, while performance is more important. We will train a model offline, which means that there is no constraint in how long it may take to train a model. This means that we will

train both a neural network, which is denoted as deep learning in the table above, and the gradient boosted decision trees based upon the strategies that are outlined in Chapter 4.

6.2 Data preparation

This section describes the preparation of data for the machine learning algorithm. The algorithm takes input as the life state of the production line, and predicts the output value. As we described in Chapter 5, the output consists of the production and weight revenues.

Data is prepared for the machine learning algorithms. This means, that all numerical data is transformed to a value between 0 and 1. The following formula normalizes the variables.

$$ScaledValue = \frac{CurrentValue - min}{max - min} \quad (6.1)$$

This way, the values of the numerical variables are scaled, while keeping its proportional relationships. As a second step, the categorical variables are one-hot encoded. For example, item codes are transformed to numerical values in this way. One-hot encoding (“OneHotEncoder”, n.d.) is the process of transforming categorical data to numerical values. Table 6.2 shows how the result of one-hot encoding. This is done for all categorical variables, which are the current item, next item and scenario number.

SampleID	Product1	Product2	Product3
Sample1	1	0	0
Sample2	0	1	0
Sample3	0	0	1

Table 6.2: one-hot encoding

6.3 Cross-validation

In order to prevent overfitting, a further process is cross-validation. This is done using a split of the trainings dataset. The model is trained on each different trainingset and evaluated on the remaining dataset, the test set. This is done multiple times, to prevent the model from overfitting.

In cross validation (“3.1. Cross-validation: evaluating estimator performance”, n.d.), the dataset is divided into multiple groups. In each iteration, each group is used once for testing, while the remaining will be used for training. Each iteration, another group of data is used for testing. In this case, the performance of the model over data that is not seen during training is evaluated. This prevents overfitting and makes use of all available data. Using this method, the MSE is calculated. The following table shows the working of this cross validation when a dataset is split into 4 subsets.

Set	Set1	Set 2	Set 3	Set 4
Iteration 1	Testing	Training	Training	Training
Iteration 2	Training	Testing	Training	Training
Iteration 3	Training	Training	Testing	Training
Iteration 4	Training	Training	Training	Testing

Table 6.3: Cross validation

6.4 Prediction functions

The algorithm we describe in this thesis uses prediction functions. These prediction functions calculate the expected reward, and are therefore important in both selecting the next context to evaluate as well as the action to evaluate based on this context. Also, these functions should be able to provide correct advice after the training phase. As also mentioned before, this is not always possible with a simple representation, but may require a more extensive function. The selected methods are a neural network and the gradient boosted decision trees. We explain the working of these functions in the following subsections.

Both models take the current state of the line as input. For both models, we normalize the variables, and hot-one encode the categorical variables. The input variables are provided to the prediction function after data preparation. This means that the input consists of the following variables:

variable	Type	Variable	Type	Variable	Type
Speed	Num.	MC1 failure time	Num.	MC4 curr. item	Cat.
Buffer level	Num.	MC2 failure time	Num.	MC5 curr. item	Cat.
Scenario	Cat.	MC3 failure time	Num.	MC6 curr. item	Cat.
MC1 tons to go	Num.	MC4 failure time	Num.	MC1 next item	Cat.
MC2 tons to go	Num.	MC5 failure time	Num.	MC2 next item	Cat.
MC3 tons to go	Num.	MC6 failure time	Num.	MC3 next item	Cat.
MC4 tons to go	Num.	MC1 curr. item	Cat.	MC4 next item	Cat.
MC5 tons to go	Num.	MC2 curr. item	Cat.	MC5 next item	Cat.
MC6 tons to go	Num.	MC3 curr. item	Cat.	MC6 next item	Cat.

Table 6.4: Sample input

During both the training and the usage of the tool, the prediction functions take the sample of Table 6.4 as input, and provides a prediction of the output as value of the trained objective value, which consists of the value of production minus penalty costs of waste.

Each model is trained using a part of the training set. Validation is done after training is completed. In order to test the performance of both models, we use the remainder of the training set as test set. We follow the cross validation procedure as explained in Section 6.3.

During training we optimize the parameters within the model to be able to provide the best prediction performance on the training set. We calculate the mean squared error using cross-validation. The mean squared error in this case means the deviation between prediction and the target value. We use the mean squared error to compare the different models and strategies.

Section 6.4.1 describes the working of the gradient boosted trees. This was the algorithm that performed best in terms of error of prediction in the data analysis in Section 6.1. As mentioned in Section 3.5.2, a neural network has been proven to be a good function in learning non-linear data relationships. Therefore, Section 6.4.2 describes the working of a neural network in the contextual, multi-armed bandit problem definition.

6.4.1 Gradient boosted decision trees

This section describes the training and set-up of the predictive function by using a gradient boosted decision tree regression function to make predictions of the target value given an input sample \mathbf{x}_i .

Each time a batch of samples is collected and presented to the machine learning algorithm, the algorithm is retrained, based on some hyper-parameters. The hyper-parameters are the

number of estimators, maximum depth, minimum samples per split, minimum samples per leaf, maximum number of features, sub-sample and learning rate. We tune these parameters to achieve the best performance. These hyper-parameters are kept the same during training, and are tuned in the end to achieve the best performance possible.

6.4.2 Neural Network

This section describes the construction of a neural network as algorithm for predictive machine learning.

The neural network consists of multiple layers, and each layer consists of a certain amount of nodes. By using dropout, some uncertainty is measured within the decisions made by the neural network. This can be used to accommodate the sampling method, which will try to decide on the best action to take. Because of the uncertain behaviour of the simulation model, not all decisions are trustworthy, meaning that some randomness should be incorporated in the neural network. By using dropout, some neurons are switched off. This enables the neural network during training to overcome over fitting and incorporate some uncertainty. This will help in sampling the best action to take.

The output of the neural network is one prediction. However, due to the use of dropout, this is different each time. Therefore, the neural network can make multiple predictions on each sample. This results in a mean and standard deviation which can be calculated per sample. Using the Central Limit Theorem (CLT) (Ganti, 2022), when there are enough observation of the same sample, the distribution of the mean will be assumed to be normally distributed. Using this fact, a confidence interval can be constructed. Using this confidence interval, the upper confidence bound can be calculated. The prediction with the highest mean will be selected as the best action to take, and the tool will give that as advice to the operators along the line.

The layers of the neural network are created where the deep layers have “relu” activation functions, and the input and output layer have linear functions. The relu activation function is the rectified linear unit, and has the following formula:

$$f(x) = \max\{0, x\} \quad (6.2)$$

The neural network consists of several layers, an input layer, an output layer and, possibly, hidden layers. Each layer consists of a number of nodes. These nodes are the number of items given as input to a layer. In this case, the input layer has 55 nodes, intermediate layers have 50 nodes, and one output node is given. The network only needs to provide one prediction.

6.5 Conclusion

In this chapter, we discussed the preparation of data in combination with the use of the predictive functions.

Data is first prepared using techniques as one-hot encoding and normalization. This will help the model to improve prediction accuracy. Both models take a sample as input, and provide one prediction as output.

The GBDT algorithm uses for this the individual trees as estimators, while the Neural Network consists of multiple layers, with nodes that are input and output to the model.

Due to the uncertainty in the machine learning models, both calculate the best action by taking multiple samples. In the case of GBDT, all estimators make a prediction, and the mean of those predictions is given as final prediction. In case of a neural network, the nodes in the layers are

trained, and using dropout, uncertainty is handled in the model. After enough predictions of the Neural Network, we calculate the mean of these predictions and take the highest as advice to the operators for implementing along the real line.

Chapter 7

Experiments

In the previous chapters, we constructed an algorithm and described methods to train a model that provides advice to operators. The model provides requested advice for line speed and buffer level ranges, where machines should switch on or off. In this chapter we describe experiments on this models, and evaluate the performance of the different settings and strategies. Therefore, this chapter aims to answer the following set of research questions:

6. *How does the model perform?*

- How does training influence the performance of the model?
- Can parameters be changed to increase performance?
- What is the effect of the model on measured output?
- How much impact does the model have on the production of waste?

This chapter further describes alternative designs. We explain the setup of experiments in Section 7.1. We explain the results of the experiments in Section 7.2.

7.1 Experimental setup

In this section we explain the experiment setup. This consists of a description of the implementation of the prediction functions, implementation of the sampling strategy and the validation procedure. In Section 7.1.1 we explain how the gradient boosted decision tree ensemble is constructed. In Section 7.1.2 we explain the construction of the neural network.

There are several input parameters that define the simulation run. The settings of the amount simulation runs and total run length can be found in Table 7.1.

The run length is taken as 1 hour because then the simulation shows what is the immediate effect of a speed change in a short period. A longer period would mean that the state has changed too much, resulting in the fact that other decisions should be taken during this evaluation time. A shorter period would be less desirable as well. The effect of a speed change is only measured after 20 minutes. In order to get a representative evaluation of the production line, some time after these 20 minutes is needed. Therefore, we take a runtime of one hour in simulation time.

In Table 7.1 we provide an overview of some settings which are relevant for the experiments. The run length is used for the simulation model. This represents the time over which the simulation calculates the amount of products produced and wasted. The number of replications is the amount of similar replications are repeated. To be able to calculate the objective value, the

revenue of production and cost of waste per ton are fixed to the values that producing a ton of waste costs three times as much as the production of a ton of production.

Because running time plays an important role in the performance of the model, besides accuracy of the predictions, we provide the settings of the neural network libraries and the type of laptop that we used to run the experiments.

Total Run Length	1:00:00.00
Number of replications	10
Revenue of production	€ [REDACTED]
Cost of waste	€ [REDACTED]
Laptop CPU	Intel Core i7-7500U
Neural Network library	Tensorflow.keras
GBDT library	sklearn

Table 7.1: experiment configurations

We distinguish different experiments. First of all, the algorithms we explained before are both tested. We train both a Neural Network and a gradient boosted decision tree algorithm. For this training, we consider two strategies for sampling trainingdata, ϵ -greedy and the softmax method. Both we test and compare for both algorithms.

We test the reaction of the machine learning algorithms when multiple simulation runs are used to get a more reliable result of one observation. We experiment with both 1 and 10 simulation runs per sample.

We validate the models against the simulation model where time is no constraint. We run the simulation model with more replications per speed for a predetermined sample, and compare the outputs of both the machine learning algorithms and the simulation model.

We validate also against using line settings or not. These line settings, so-called scenarios, determine at which buffer level the machines should be switched on and off. In the setting with line settings, we use an equal weight between the different types of products produced, showcases and multipacks. In the situation without line settings, these weights become more important, because the products with higher priority are assigned higher weights.

The following subsections describe the setup of the machine learning algorithms. Section 7.1.1 describes the GBDT and Section 7.1.2 describes the Neural network.

7.1.1 Gradient boosted decision trees

The gradient boosted decision trees are implemented in Python. The library used for this algorithm is sci-kit learn (Pedregosa et al., 2011). It uses the gradient boosting regressor package. It takes as parameters a loss function, which is set to “negative mean squared error”, and a number of estimators. Because of the negative mean squared error, the aim is to maximize this error and get it as close to zero as possible.

The other estimators are the number of trees that are constructed. The gradient boosted decision trees are trained using a dataset that is collected by the contextual bandit method. The hyper parameters are fitted according to a grid search on an initial model, where the performance of each parameter is evaluated, and the best is chosen. This resulted in the parameters that we show in Table 7.2.

Hyperparameter	Value (incl. linesettings)	Value (excl. linesettings)
Nr. of estimators	90	90
Learning rate	0.05	0.05
max depth	15	9
min samples split	50	190
min samples leaf	20	10
max features	39	49
subsample	0.85	0.75

Table 7.2: Hyperparameters GBDT

7.1.2 Neural network

The neural network uses the library tensorflow (Martín Abadi et al., 2015). From tensorflow, keras is imported as package to python. There are 2 hidden layers, 1 input layer and 1 output layer. The input layer has the size of the number of variables that is given as input, which is 56 in total. The hidden layers have each 50 nodes, and the output layer has 1 node. This output layer provides the prediction of the neural network. Here, the loss function is “mean squared error” as well, while the optimizer is “SGD” (stochastic gradient descent). Dropout is used to switch neurons off during training, which helps prevent the model from over fitting. Dropout is fitted on the second and third layer, with a percentage provided in Table 7.3.

Parameter	Model incl. line settings	Model excl. line settings
Optimizer	Stochastic Gradient Descent	Stochastic Gradient Descent
Dropout	0.25	0.25
Layer nr.	Nr. of nodes	
Layer 1	50	45
Layer 2	40	35
Layer 3	40	35
Layer 4	1	1

Table 7.3: Neural Network settings

7.2 Results

In this section we present the results of the experiments that are defined in the previous Section. In Section 7.2.1 we describe the performance of all experiments in terms of Means Squared Error. In Section 7.2.2 we discuss the results of the different methods of sample selection to create a training set, while we discuss the difference in input in the simulation model in Section 7.2.3.

7.2.1 Machine Learning algorithm performance

To be able to calculate the performances of the individual models, the training set is divided into a training set and a test set. The models are trained using this training set, and the performance is evaluated using the predictions of the model on the test set. The difference between these predictions and the true results are calculated in terms of the Mean squared error (MSE). Also, the number of samples and strategy of sampling is given.

We used the experimental settings in Table 7.1 to perform these experiments. Per algorithm, we trained using the number of samples as mentioned in Table 7.4, based on a sampling strategy. Also, we took different amounts of replications from the simulation model to calculate the

reward which is given as input to the Machine Learning algorithm. We stopped training after the number of samples that is given in the table.

ML algorithm	Strategy	MSE	# Samples	Sim. replications
Neural Network	Softmax	0.0091	5500	1
	Softmax	0.0061	2300	10
	e-greedy	0.0024	2500	10
GBDT	Softmax	0.0027	5200	1
	Softmax	0.0013	5500	10
	e-greedy	0.0013	3000	10

Table 7.4: Results of experiments

We notice that the Mean squared error in case of the Gradient boosted decision tree is always lower than its corresponding Neural Network variant. Also, we see that increasing the sample size does not improve the performance of the GBDT after a while. Also, for the Neural network we do not see an improvement when the number of samples increase.

We also tested on the calculation time, where we took the average of the prediction over 100 random created samples. The results are that the GBDT took on average 0.0044 seconds per sample, while for the Neural Network, this took 10.71 seconds. For both models, this is a allowed calculation time, but we notice that the prediction time for the GBDT is faster.

7.2.2 Sample selection

In this section we describe the experiments regarding sample selection. We use two strategies, which are explained before in Chapter 4.

We compare the performance of the Boltzmann function to select samples, together with the ϵ -greedy function. For both sampling strategies we use a scheme that adjusts a temperature according to the desired exploring or exploiting strategy. In the case of Boltzmann function, we update rho towards a higher value. When rho becomes high, the probability of exploitation increases. For e-greedy, we use alpha. Alpha is adjusted from 1 towards zero, where alpha represents the probability of exploration. In the beginning, this is large, and decreases as the number of samples increases. This results in more exploitation when training time increases, but also makes sure that in the beginning exploration is used to get more insight about the true value of the samples.

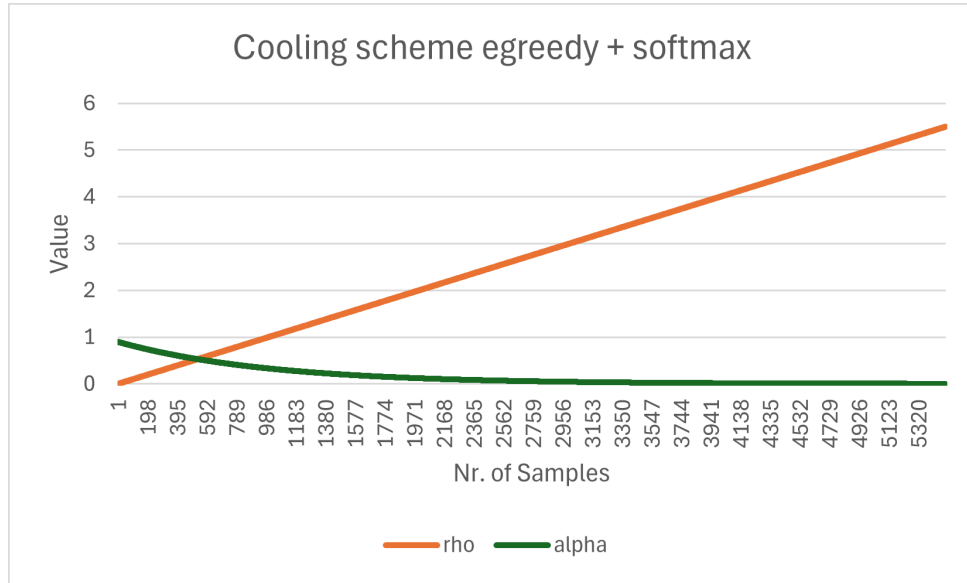


Figure 7.1: Cooling scheme

Figure 7.1 presents both values of ρ and α per iteration. For this cooling scheme we used the following updating formulas.

$$\alpha_t = \alpha_{t-1} * 0.999$$

where $t > 0$ and $\alpha_0 = 0.99$ and

$$\rho_t = \rho_0 + 0.001 * t$$

where $t > 0$ and $\rho_0 = 0.001$.

We compare the performance of both strategies for both Neural Network and Gradient Boosted Decision Trees, where we compare the mean squared error over the number of samples in the training set. Figure 7.2 and Figure 7.3 compare the strategies for the GBDT and NN respectively.

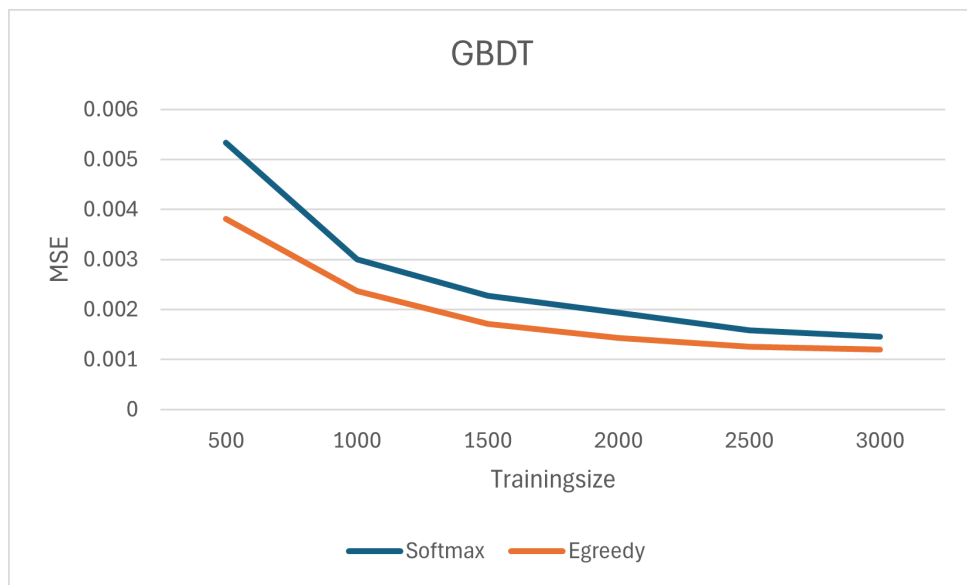


Figure 7.2: GBDT strategy comparison

We see in Figure 7.2 that both models MSE decreases when the number of samples in the training dataset increases. However, the ϵ -greedy is lower for all sizes of the training set. As we saw in the final MSE in Table 7.4, both MSEs converge to the same value as soon as the number of samples keep increasing. This means that for the GBDT, taking ϵ -greedy in the selection of samples is better when there is a short time available for training, but does not really improve the model when more training time is available.

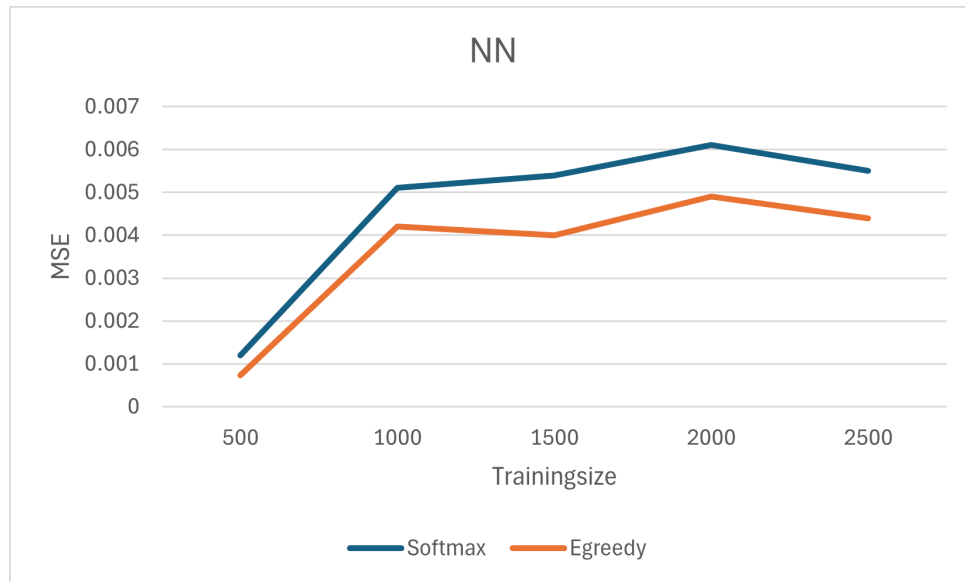


Figure 7.3: NN Strategy comparison

Figure 7.3 shows the performance of the Neural network over time when sample sizes increase. We show the performance in terms of the Mean Squared Error per size of the training set. We notice that the size of the training set is not important for the neural networks performance in terms of the MSE. This remains stable as soon as the training set reaches a value of 1000 samples. The 500 sample estimate here is not a reliable result, because the training set and validation set has equal size.

The strategy of using ϵ -greedy in this case is better for smaller training durations than the Softmax duration. This can be due to the cooling scheme used in both scenarios.

For smaller trainingsets, we see that using ϵ -greedy is a better strategy to get a smaller mean squared error. When there is more training time available, the difference between both strategies becomes smaller.

7.2.3 Simulation

In this section we describe the experiments that are performed with changes in the simulation model. We perform experiments where one replication of a sample is used, and with 10 replication. This section describes the results of both algorithms with this input. We compare the results in terms of mean squared error on the training set, and view the difference between 1 and 10 iterations of the same sample by the simulation model.

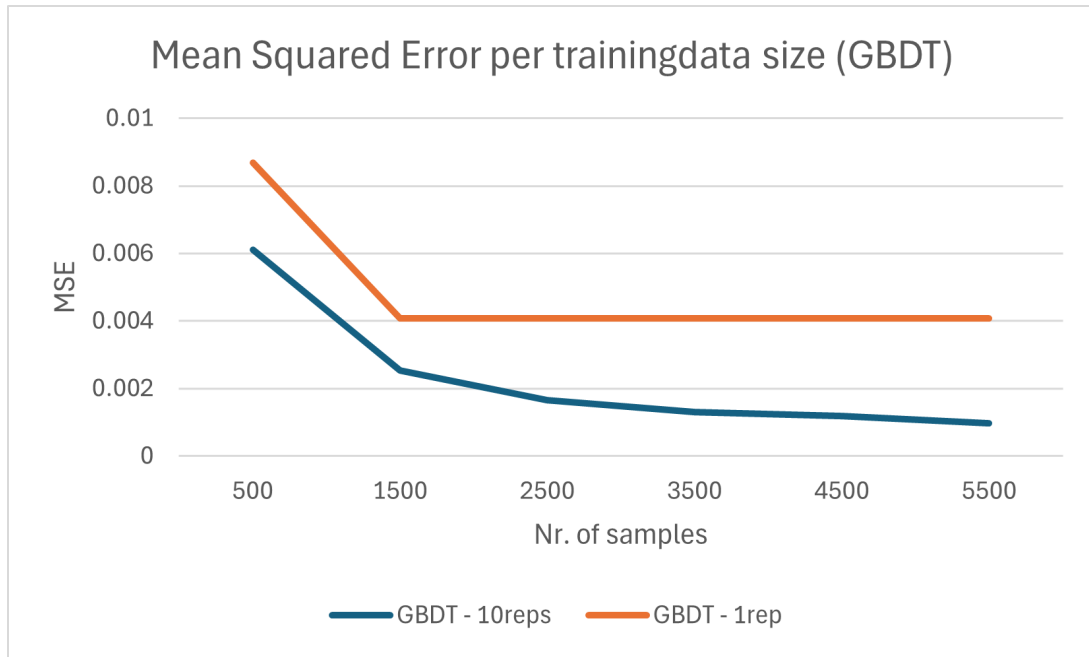


Figure 7.4: MSE to sample size comparison GBDT

Figure 7.4 shows that the MSE for ten samples is in every point of estimations lower than the MSE for 1 observation per sample. This would mean that the performance of 10 replications is outperforming one replication per sample in every case.

In the case of the GBDT, the MSE does not decrease anymore from 1500 samples onwards when only one replication is used. In the case of 10 replications, the MSE keeps getting lower as more samples are added to the trainingset. This means that using 10 replications per sample, and taking the average as input for the trainingdata, increases the model performance.

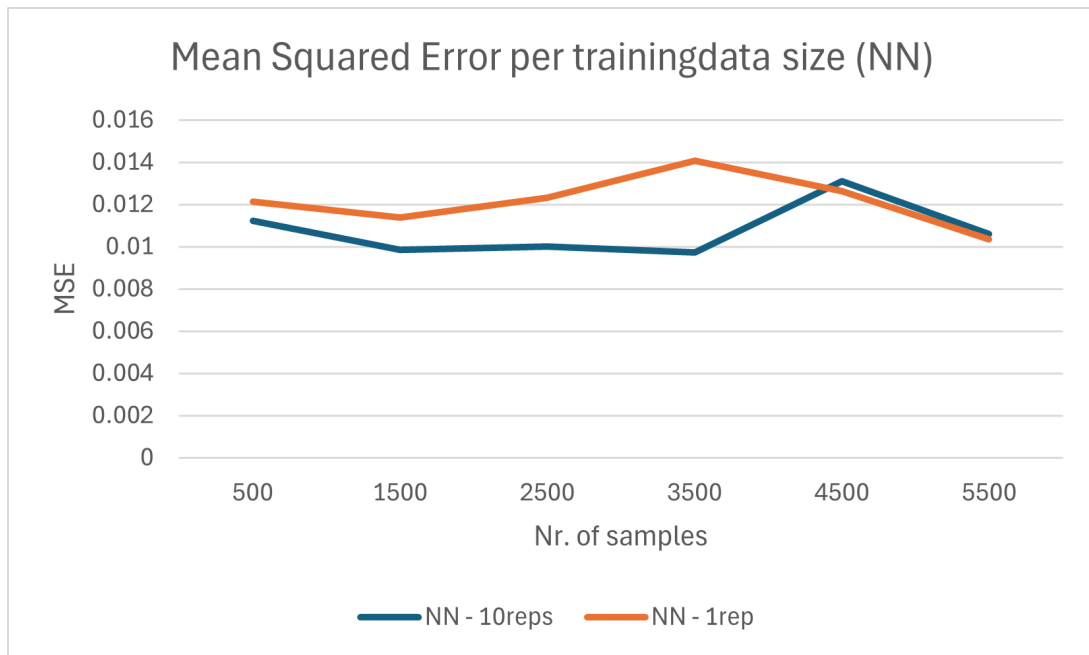


Figure 7.5: MSE to sample size comparison NN

In the case of the Neural Network we observe that the MSE stays stable, does not decrease with

more samples, but seems to be lower with 10 replications than with 1 replication. This would mean that also in this case the use of 10 replications is better than 1 replication, especially when the training time is limited.

We notice that using more replications of the simulation model in the training period is helping both models to overcome the uncertainty. This can be due to the large variation in performance of the production line. This means that this randomness is a little less using more replications. This means that using more replications for the same sample, and taking the average as output of the simulation model improves performance of the machine learning algorithms. However, the trainings time increases significantly. Instead of 5500 times running one replication, now we run 5500 times 10 replications. Training is done offline, so this is not really an issue, but this is a point of consideration.

During training we notice that the variation of one sample can have a large variation between two random observations. This means that in one replication, lots of waste can be produced, while in another replication, with the same settings, a high reward is observed. To overcome this problem, and to help the machine learning algorithms better understand the dynamics of the input variables against the output variable, we run the simulation model with 10 replications, and take the average of this. We provide this value as result to the machine learning algorithm. In this case, the performance of both algorithms increase, and training the algorithms takes less time.

7.2.4 Advice creation

The next experiment we perform is about the time between two advises. We know that the time to fully notice the change of speed is 20 minutes, which means that changing the speed faster is not really desired. We set up this experiment with an 8 hour run. This run is performed for 100 replications. During each interval of 15, 30, 45 or 60 minutes, the simulation stops and requests advice to the machine learning algorithm for implementing a new speed. This advice is created and implemented in the simulation model. We measure the average over all replications, and depict the results in Figure 7.6.

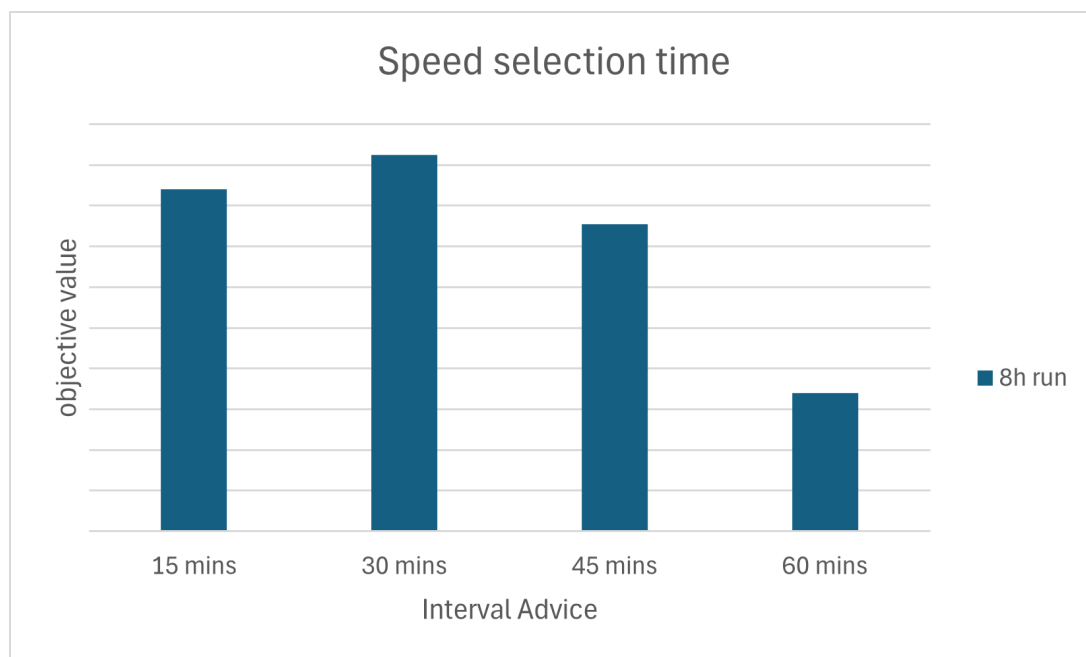


Figure 7.6: Comparison of time between advice

Figure 7.6 shows the results of implementing advice in certain intervals, based on the best working GBDT model. This model is trained with ϵ -greedy strategy on 5500 samples. We see that implementing advice each 30 minutes creates a higher reward than both shorter and longer. This results show that implementing advice for a shorter time interval is not necessary, and does not add more value, while waiting a longer period of time might mean that the buffer will overflow, due to a too high speed, or that there is more opportunity to increase speed, which means that more revenue can be obtained from changing the speed. This means, that implementing advice each 30 minutes is most valuable for the production line.

7.3 Validation

In this section we provide an analysis of the performance of the machine learning algorithms compared to the simulation model. We select two samples that have a different best speed in output. We run the simulation model on all speed possibilities. In this validation, we assume that time is no constraint, meaning that the simulation model can run as long as necessary such that we are able to calculate a significant result.

The aim of this research is to create a tool that can predict accurately the next optimal settings. This section aims to validate the performance of the functions regarding this goal. We compare our experiments to the samples, and review the differences between the simulation output and the output of the machine learning models. Table 7.5 and Table 7.6 show the two samples that we provide to the simulation model to calculate the objective value. We only test on two samples because of the long running time of the simulation model on one sample.

Speed		Scenario	4			
	CurrItem	status	failure time	buffer	nextitem	progress
Mc1	24x3	Failed	178		18x6	30
Mc2	5x32	Running	0		5x32	8
Mc3	18x4	Running	0		26x10	30
Mc4	1x32	Running	0		1x30	15
MC5	18x12	Running	0		24x7	68
Mc6	1x30	Failed	234	38	10x25	20

Table 7.5: Test sample 1 with linesettings

Table 7.5 shows a sample where two machines are failing, but the failure time is not very high. Also, there is not much progress left at every machine. The buffer is also not filled very much. This means that the expected value of the optimal speed is around \blacksquare . We use this sample to see if the machine learning algorithms can provide a good advice when not much is happening at the production line, and should therefore recognize these situations to provide advice that is corresponding to this situation.

Speed		Scenario	7			
	CurrItem	status	failure time	buffer	nextitem	progress
Mc1	24x3	Running	0		12x7	93
Mc2	5x32	Failed	882		1x32	78
Mc3	18x6	Failed	1800		18x6	28
Mc4	5x32	Failed	3600		1x32	66
MC5	18x12	Failed	3600		24x7	57
Mc6	1x32	Running	0	90	10x25	5

Table 7.6: Test sample 2 with linesettings

Table 7.6 shows the second test sample we use to validate the machine learning models with the simulation model. In this situation, more machines have already failed, and for a longer time period. This means that probably, these machines will be in failure for a longer time. This means that the speed needs to be adjusted to that, and this will result in a lower speed. The aim of this sample is to see how the machine learning algorithms cope with this situation and will provide advice on a situation where adjustments are needed to prevent too much waste production.














































speed	Sample 1	Sample 2
		
		
		
		
		
		
		
		
		
		
		
		
		
		
		

Table 7.7: Simulation results Sample 1 + 2

From Table 7.7 we see that in case of sample 1, the optimal speed is 100 cuts per minute, while in sample 2, the optimal speed is 100 cuts per minute. The value of production minus waste is highest at these speeds. For this calculation we constructed a 95% confidence interval. The results of this confidence interval is added to Appendix D. We compare the models of the neural network and the gradient boosted decision trees with the results of the simulation model.

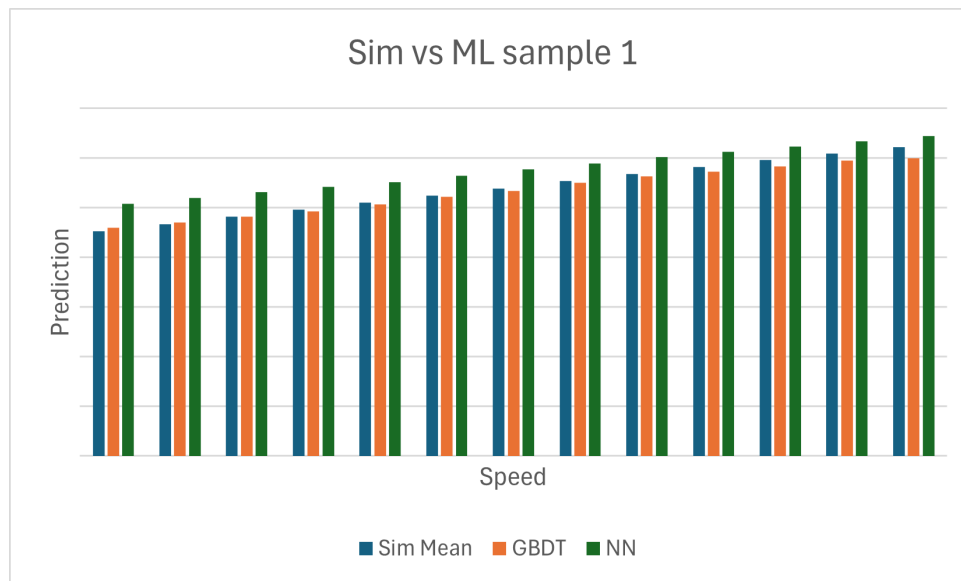


Figure 7.7: Results Validation Sample 1

From Figure 7.7 we see that the GBDT has been better able to predict the output for every speed. We see that the GBDT model follows the pattern that is also found in the simulation model. We notice however that the Neural Network has less been able to follow this pattern. However, both algorithms predict ██████ as optimal speed. We see that the Neural network has more deviation from this real predicted value than the GBDT.

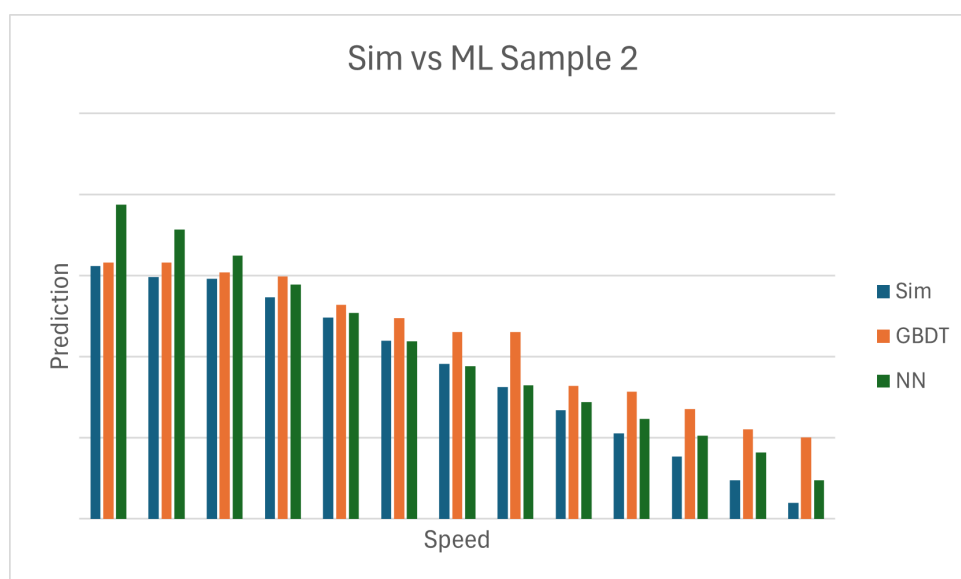


Figure 7.8: Results Validation Sample 2

In Figure 7.8 we see that again the GBDT follows the pattern of the simulation model, while the Neural Network also recognizes ██████ as optimal speed, but does at all times predict the output too high. The GBDT however predicts too low on performance as soon as the speed increases to ██████. We see that both algorithms were able to predict the correct speed, however the GBDT algorithm was more close to the real prediction output, especially around the optimal speed selection than the Neural Network.

We see from this part of the validation that the model is able to perform good on the prediction

in two instances. However, this is not reliable for the total sample space. In order to test this, we run 50 samples. The simulation model provides a solution on bigger step sizes of each of these 50 samples. To reduce the running time of the simulation model, we take 100 replications of stepsize 100, which means that only for speeds divided by 100 are evaluated. Table 7.8 shows the ranges of speed evaluated by the simulation model. When the machine learning model predicts the speed within the range corresponding to this speed, we consider the prediction correct.

We use only 50 samples due to the long calculation time needed for the simulation model to evaluate all these samples and provide a correct output after sufficient amount of iterations. For the same reason, we use a step size of 5. With these settings, we can construct confidence intervals that provide statistically significant results about the true mean.

Simulation speed	Correct range
100	100 - 100
200	200 - 200
300	300 - 300
400	400 - 400
500	500 - 500

Table 7.8: Validation speed ranges

We let the machine learning model calculate the optimal speed for each of the 50 samples. When the prediction is within the ranges of Table 7.8, we count it as a correct prediction. Otherwise, we count it as a wrong prediction. Table 7.9 show the results in terms of percentage of correct selection for each model. We see that the GBDT model performs better in this selection procedure, with overall higher percentages than the Neural Network. It shows a confirmation of the results we have seen before, where the GBDT model is more capable of recognizing the correct situation.

GBDT	Percentage	NN	Percentage
egreedy (5500)	78	egreedy (5500)	74
softmax (5500)	86	softmax (5500)	76
egreedy (3000)	76	softmax (3000)	76
softmax (3000)	86	softmax (1rep)	66
softmax (1rep)	72		

Table 7.9: Validation results

From the best performing model, we analyze what would happen when every variable remains the same, but the buffer levels increase, or the machine failures change. Both variables have impact on the advice that is given by the model, and therefore we want to see what values of these variables have impact on the speed advice.

We show the effect of the buffer level in Figure 7.9. We change the fill rate of the buffer, but keep the rest of the input the same. We request advice on each of these buffer levels regarding speed. We analyze two scenarios, one where there is no machine failure yet, and one where machines are failing. We see that the optimal speed remains 100, independent of the buffer level, when no machines are failing at the moment of decision making. However, we also see that when machines are failing, the speed is reduced in the beginning, and further reduced as soon as the buffer is filled more. This means that the chance of producing waste is increasing, and the speed should be reduced.

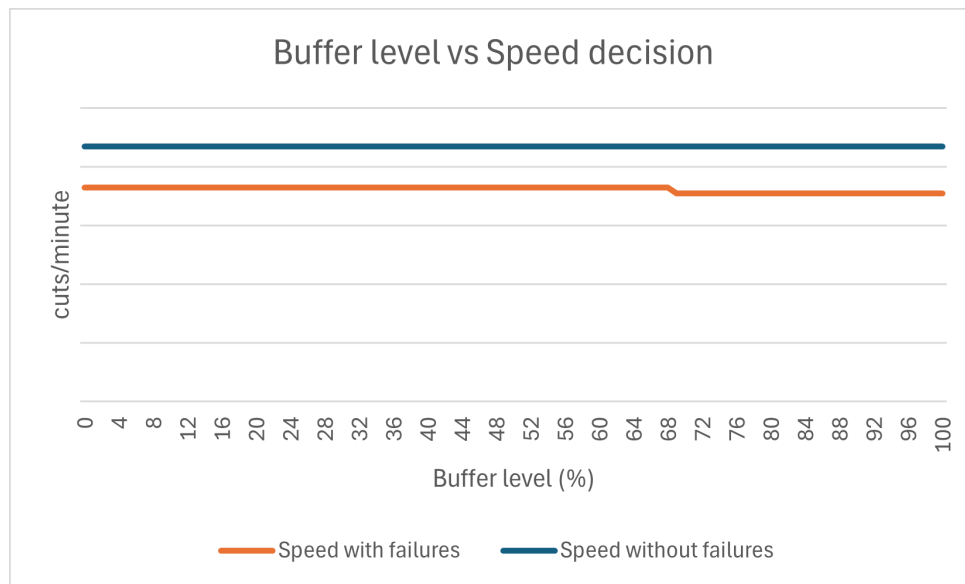


Figure 7.9: Analysis buffer level vs speed

Figure 7.10 show the predicted optimal speed given the machine failures. The graph shows the failure times of machines in seconds, with the optimal speed corresponding to the total failure time. The buffer is almost empty, to leave out its influence on the decision. We see that as the machines' failure times increases, the speed should be reduced. As soon as multiple machines are failing a longer time, the speed is drastically reduced. In sample 14 for example, only 1 machine is failing, which means that the speed can be increased again. We see that as soon as two machines are failing for a longer period, the speed reduces to the minimum speed possible.

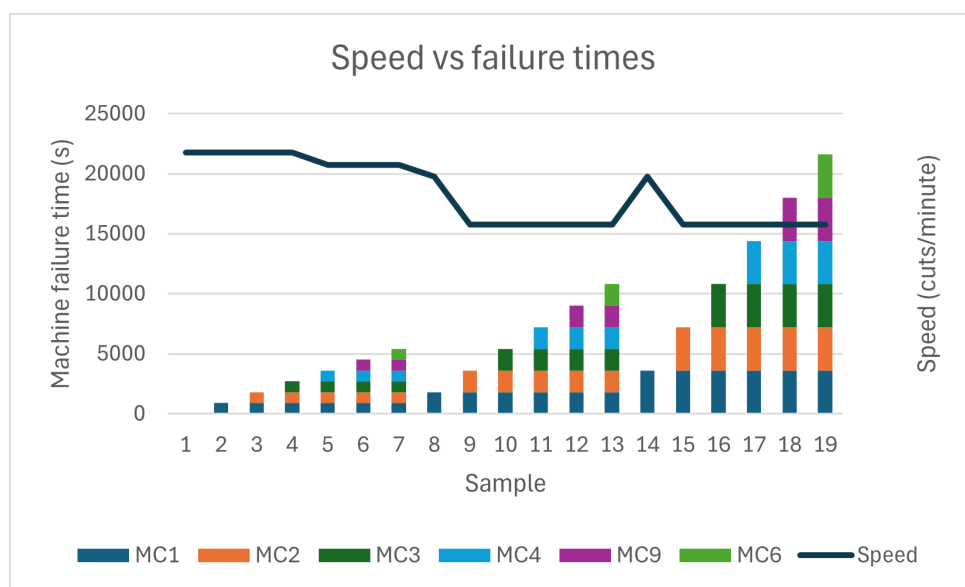


Figure 7.10: Analysis speed vs machine failure time

When running without linesettings, we run the simulation model once more, on the sample in Table 7.10. We compare the performance of this simulation model and machine learning algorithms.

Speed						
	CurrItem	status	failure time	buffer	nextitem	progress
Mc1	18x5	Running	0		18x5	14
Mc2	1x32	Failed	402		1x30	3
Mc3	26x10	Failed	130		36x3	20
Mc4	1x32	Running	0		1x30	55
MC5	24x7	Running	0		24x7	28
Mc6	5x32	Running	0	6	5x32	30

Table 7.10: Test sample without line settings

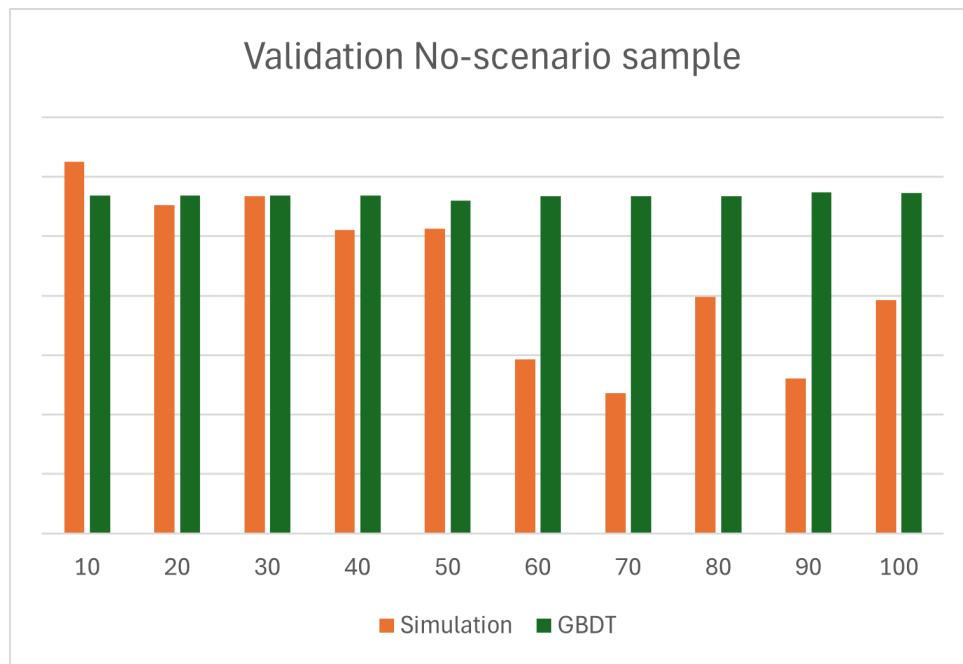


Figure 7.11: Validation sample without line settings (speed=)

Figure 7.11 shows the validation of the GBDT model and the simulation model. The objective is changed to optimize the value where either multipacks or showcases are prioritized. The machine with the lowest priority, calculated based on the formulas in Chapter 5, are used to determine at the begin of the run which machine to switch off. The goal of the machine learning model is to provide an optimal speed, and an optimal buffer level which should determine when these machine should be switched off.

The results are summarized in Figure 7.11, where we see that a small buffer level is optimal, but that the real values do not change that much in the simulation model. We see that the machine learning algorithm failed to notice the change between the buffer levels. This could either be due to more randomness in the simulation model, or a small change in output given this buffer level.

7.4 Conclusion

This chapter describes the experiment setup and results. We trained two machine learning algorithms using a simulation model. We tried different experiments regarding sampling strategy and simulation observations, and validated the model against this simulation model as if time

were no constraint.

We conclude that the epsilon greedy method works better in all cases where less training time is available. When more training time is available this has a smaller effect, meaning no difference is seen between the two strategies. For both models this holds. We also see that using one simulation observation, variation in the machine learning model increases significantly, ensuring that using more simulation observations, and taking the mean increases model performance.

We validated against the simulation model, where the simulation model received enough time to calculate a good result. We compared the models against two samples. In both scenarios we saw a good performance, where all models predicted the output correctly. The GBDT seems to predict more accurate around the real value.

In case the line settings are removed, and the model should evaluate which buffer levels should be used to switch machines off and on, we notice that the difference between the options is very small. This could have a couple of reasons. A reason can be that the dataset of training is too small, or that the differences between the options is too small. The machine learning algorithm cannot decide between the difference of two options.

Chapter 8

Conclusion and Recommendations

8.1 Conclusion

In this thesis, we described the problem that the speed and line settings of the production lines are unknown. Operators along the production line do not know the correct and optimal speed. The speed determines the rate at which products enter the machines, while line settings determine which machines can be switched on and off. The factor that impacted this were the unknown cause of machine standstill, which is caused by machine failures and changeovers of packaging materials.

In order to solve this problem of no optimal policy of line configurations, we created a decision making tool, to help the operators along the line in adjusting their line configurations. Literature showed that for short-term line adjustments, machine learning could help, in combination with a digital twin. Contextual bandits can be used to train a predictive algorithm, which can provide advice to the operators. Using this digital twin simulation model can increase training speed, and provides more flexibility in the creation of training data.

For training, the real line cannot be used. During training, also the effects of non-optimal speeds could be advised, which means that this disrupts the production. As alternative, a replacement should be created. For this, the digital twin in the form of a simulation model is used to predict the performance of the real-life production line. With this prediction, data can be gathered to train the predictive algorithms.

Therefore, the chosen solution method is making use of the contextual bandit approach. This approach selects what action to take in order to learn as much as possible. This action selection is based on a sampling strategy. This sampling strategy provides the policy which action to select, and feed to the simulation model to get a reward.

The simulation model looks one hour ahead, to predict production quantities and waste quantities per sample. Per line configuration, the simulation model provides a value of the objective function, which is a weighted difference between production and waste.

Using these different sampling strategies, the training set is created as efficiently as possible. Only samples where the potential reward is high are added to the training set and the real reward is asked from the simulation model.

After normalization and other data preparation techniques, the predictive functions are trained. From both literature and data analysis, we created a Gradient Boosted Decision tree regressor and a Neural Network regressor. Both models are trained on input data, and we aimed to let them predict the reward of this input data.

Based on the mean squared error, which is calculated on a part of the trainingset, we see that the gradient boosted decision trees in general performs better. In all situations, the MSE is lower than the neural network. This means that the predictions of the GBDT is closer to the real expected values of the simulation model than the estimates of the neural network.

When comparing the sampling strategies, the ϵ greedy sampling strategy performs better than using the Boltzmann function. This difference is more noticeable when the training set is smaller. So, when less training time is available, the ϵ -greedy function is a better strategy than the Boltzmann. As soon as the training time is no constrained anymore, both strategies seem to work equally well.

In validation with the simulation model, we see again that the GBDT is better able to predict the real performance. On both samples where line speed is high and where line speed is low, the algorithm provides good predictions towards the optimal speed. The Neural Network seems to suffer with these predictions a bit more. This is also reflected in the Mean Squared Error. The Neural Network overestimates in both samples the performance, resulting in a too high prediction over all speeds.

However, both models achieve to predict the correct speed in both situations. This means that we would be able to use this model to predict the speed and advice operators along the production line about the best speed to run for the next period.

When we include the line settings in the action set of the agent, we see that there is no difference between the Mean Squared error which is calculated. We notice a good prediction performance over the validation set. We however also conclude that there is no large difference noticeable in the simulation model between the different options, making it difficult to create a good advice. This also is reflected in the performance of the machine learning model, making it impossible to train this properly.

The tool eventually can provide advice about speed to the operators, which results in better performance regarding waste and production balance. The speed is predicted such that this balance is optimal, regarding the values of producing waste and revenues of producing products. We saw that in 86% of all situations, the prediction was close enough to the real value in the best performing model. This means that when only advising about the speed, Table 8.1 shows what model has the best configurations.

Setting	value
Advice interval	30 minutes
Algorithm	GBDT
Sampling strategy	ϵ -greedy
Nr. of samples training	5500

Table 8.1: Best model

8.2 Recommendations

This section describes the recommendations for the implementation of the model at the production line. Also it provides recommendations regarding the results and conclusion.

We concluded in the results section that the models, with only predicting the impact of future speed of the model, are validated and follow the patterns of the simulation model. The first step should then be to validate this model with the real life production line. The aim should be to see if the model is trained on the correct data. This means that this model provides accurate

results, with a fast runtime, which helps the operators along the production line in improving the performance of the production line.

In case of further optimization without the line settings, more research is needed to see how to increase the effect of the buffer levels. We concluded that the current way of working with different scenarios is best for now, but can be improved by dynamically adjusting these buffer levels. This should be done by analyzing the priorities of the different products. However, the difference between buffer levels should be made more important, through which the machine learning function can better estimate the impact of priority of the products.

Regarding implementation, the following plan can be followed to make sure that the tool is used accordingly. In the first stage, the tool should be run alongside the real line, where it provides advice based on the input when requested by operators. The operators should take into account the advice and accept or reject the advice. When rejected, a reason should be logged, in order to gain insight in reasons why the advice is not followed.

After this phase, the company should look, together with the operators, to the cases when the advice was rejected. The model can be retrained on these situations to improve the performance. When this works better, the advice can be followed more often by the operators, and the model is able to help the operators in deciding the next state of the production line. This means that the operators get insight in how the production line performance can be improved, and get augmented advice about how the production line settings should be changed. However, this tool is an advisory tool, and operators will still have to make the decision.

In the end, this should result towards the situation where operators receive advice from the machine learning tool. It is therefore important to take into account the operators which have to work with this tool, and review the reasons of not implementing the advice created by the tool. This should prevent that the advice is followed without reasoning, and prevents situations in which the tool does not provide the best advice.

8.3 Limitations and future research

In this section, we describe the ideas for future research and limitations of this thesis. There were some assumptions made that make the model less realistic for use in combination with the real production line, and some could be incorporated for future research. The following list provides some ideas that might be usable for either the company or a further research project to take into account and improving the corresponding decision-making tool.

First, we will discuss the limitations of this research, afterwards we discuss the points of future research.

The main limitation of this research is that the data used for training is simulation data. This means, that there is no certainty that the real production line is reacting in the same manner as the simulation model. In order to overcome this, either the simulation model should be evaluated along the production line, and if that is evaluated then the machine learning model can be evaluated. Also, if something happens to be different along the real line, the model should be retrained. Therefore, a limitation is that the simulation model should be evaluated along the real line, and reflect the performance of the real production line, in order to achieve a good performance of the model.

Another limitation of this research is that the validation is only done on two samples. On these two samples, the performance was good. However, this does not promise to be towards other samples. The reason for only validating with two samples is the large duration of the simulation model to create significant different outputs in advice. This is a limitation in the testing and validation phases of the model. In order to see how the model reacts on other samples, more

testing should be done. In order to keep running time short, we only used two samples in this research.

The following points describe the elements for future research.

1. In this thesis, we use a simulation model to train the machine learning algorithm. However, it is time costly to construct such a model, which should be validated along the real-life production line. Therefore, it might be beneficial to spend research on whether an algorithm can be trained using real life data, which tries to learn the behaviour of the production line in an as optimal as possible way.
2. The failures of the machines along the packaging line is taken, due to limited available data, as a single variable, where no relation between different types of failures is taken into account. When this is researched further, the prediction of the model might improve, because the prediction of failures and failure duration can improve.
3. The current decision is made for one hour. This is done based upon the assumption that for reality this is the most practical. For less time, there would be too much speed changes, while providing advice for a longer time horizon might mean that a change in settings might be beneficial. Therefore further research can be conducted in the direction of dynamically determining how long the current settings are beneficial, and when a new change in settings, and new advice should be provided.
4. The current model takes the production schedule as taken, which might not be the most optimal schedule. Therefore future researchers can conduct research on scheduling items on machines, to even speed up the process of producing certain product types.
5. In order to be able to compete with the machine learning model, the simulation model requires further research, in how the computing budget can be optimally assigned. Certain scenarios are more beneficial for the result to create more certainty about the result, and might require more calculation time.
6. Failure prevention can be a topic of future research. By creating a model that predicts failures more accurately, by checking the products before entering the packaging line, failures can be prevented. This can improve the performance of the production line.
7. Finally, the regulating of waste can be a topic for conducting further research. Waste is now taken as one cost on production output, but not all waste has the same cost. This might change the decisions made by the tool, and might change the way decisions are made. This can also take into account the relative amount of waste through the whole production process.

Bibliography

- 3.1. *cross-validation: Evaluating estimator performance*. (n.d.). Retrieved August 23, 2024, from https://scikit-learn.org/stable/modules/cross_validation.html#cross-validation-iterators
- Alpaydin, E. (2010). *Introduction to machine learning*. (Vol. 2nd ed). The MIT Press. <http://ezproxy2.utwente.nl/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=307676&site=ehost-live>
- Altair rapidminer. (n.d.). Retrieved June 1, 2024, from <https://altair.com/altair-rapidminer>
- Anzai, Y. (1992). 10 - learning by neural networks. In Y. Anzai (Ed.), *Pattern recognition & machine learning* (pp. 297–335). Morgan Kaufmann. <https://doi.org/https://doi.org/10.1016/B978-0-08-051363-8.50014-4>
- Auer, P., Cesa-Bianchi, N., Freund, Y., & Schapire, R. E. Gambling in a rigged casino: The adversarial multi-armed bandit problem [Cited by: 534]. In: Cited by: 534. 1995, 322–331. <https://www.scopus.com/inward/record.uri?eid=2-s2.0-0029513526&partnerID=40&md5=522cdda5087968e00b444bf303081841>
- Ayodele, T. O. (2010). Types of machine learning algorithms. In Y. Zhang (Ed.), *New advances in machine learning*. IntechOpen. <https://doi.org/10.5772/9385>
- Balef, A. R., & Maghsudi, S. (2023). Piecewise-stationary multi-objective multi-armed bandit with application to joint communications and sensing. *IEEE Wireless Communications Letters*, 12(5), 809–813. <https://doi.org/10.1109/LWC.2023.3244686>
- Benfer, M., Peukert, S., & Lanza, G. (2021). A framework for digital twins for production network management [54th CIRP CMS 2021 - Towards Digitalized Manufacturing 4.0]. *Procedia CIRP*, 104, 1269–1274. <https://doi.org/https://doi.org/10.1016/j.procir.2021.11.213>
- Biller, B., Jiang, X., Yi, J., Venditti, P., & Biller, S. (2022). Simulation: The critical technology in digital twin development. *2022 Winter Simulation Conference (WSC)*, 1340–1355. <https://doi.org/10.1109/WSC57314.2022.10015246>
- Boschert, S., & Rosen, R. (2016). Digital twin—the simulation aspect. In P. Hehenberger & D. Bradley (Eds.), *Mechatronic futures: Challenges and solutions for mechatronic systems and their designers* (pp. 59–74). Springer International Publishing. https://doi.org/10.1007/978-3-319-32156-1_5
- Bouneffouf, D., & Claeys, E. (2021). Online hyper-parameter tuning for the contextual bandit. *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 3445–3449. <https://doi.org/10.1109/ICASSP39728.2021.9413526>
- Cakmak, S., Wang, Y., Gao, S., & Zhou, E. (2024). Contextual ranking and selection with gaussian processes and optimal computing budget allocation [Cited by: 0; All Open Access, Bronze Open Access]. *ACM Transactions on Modeling and Computer Simulation*, 34(2). <https://doi.org/10.1145/3633456>
- Cakmak, S., Zhou, E., & Gao, S. (2021). Contextual ranking and selection with gaussian processes. *2021 Winter Simulation Conference (WSC)*, 1–12. <https://doi.org/10.1109/WSC52266.2021.9715499>
- Chiurco, A., Elbasheer, M., Longo, F., Nicoletti, L., & Solina, V. (2023). Data modeling and ml practice for enabling intelligent digital twins in adaptive production planning and con-

- trol [4th International Conference on Industry 4.0 and Smart Manufacturing]. *Procedia Computer Science*, 217, 1908–1917. <https://doi.org/10.1016/j.procs.2022.12.391>
- Daoutidis, P., Lee, J. H., Harjunkski, I., Skogestad, S., Baldea, M., & Georgakis, C. (2018). Integrating operations and control: A perspective and roadmap for future research. *Computers & Chemical Engineering*, 115, 179–184. <https://doi.org/10.1016/j.compchemeng.2018.04.011>
- Ferriol-Galmés, M., Suárez-Varela, J., Paillissé, J., Shi, X., Xiao, S., Cheng, X., Barlet-Ros, P., & Cabellos-Aparicio, A. (2022). Building a digital twin for network optimization using graph neural networks. *Computer Networks*, 217, 109329. <https://doi.org/10.1016/j.comnet.2022.109329>
- Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5), 1189–1232. Retrieved June 7, 2024, from <http://www.jstor.org/stable/2699986>
- Ganti, A. (2022). *Central limit theorem (clt): Definition and key characteristics*. Retrieved August 23, 2024, from https://www.investopedia.com/terms/c/central_limit_theorem.asp
- Gao, S., Du, J., & Chen, C.-H. (2019). Selecting the optimal system design under covariates. *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, 547–552. <https://doi.org/10.1109/COASE.2019.8842957>
- Ghaleb, M., Namoura, H. A., & Taghipour, S. (2021). Reinforcement learning-based real-time scheduling under random machine breakdowns and other disturbances: A case study. *2021 Annual Reliability and Maintainability Symposium (RAMS)*, 1–8. <https://doi.org/10.1109/RAMS48097.2021.9605791>
- Goodwin, T., Xu, J., Chen, C.-H., & Celik, N. (2021). Efficient simulation optimization with simulation learning. *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, 2268–2273. <https://doi.org/10.1109/CASE49439.2021.9551410>
- Hastie, T., Tibshirani, R., & Friedman, J. (2009, February). *The elements of statistical learning* (2nd ed.). Springer.
- Heerkens, H., & Winden, A. v. (2017). *Solving managerial problems systematically* (Vol. First edition). Noordhoff Uitgevers BV.
- Hong, L. J., & Jiang, G. (2019). Offline simulation online application: A new framework of simulation-based decision making. *Asia-Pac. J. Oper. Res.*, 36(06), 1940015.
- Hong, L. J., Nelson, B. L., & Xu, J. (2015). Discrete optimization via simulation. In M. C. Fu (Ed.), *Handbook of simulation optimization* (pp. 9–44). Springer New York. https://doi.org/10.1007/978-1-4939-1384-8_2
- Jäger, G. (2021). Using neural networks for a universal framework for agent-based models. *Mathematical and Computer Modelling of Dynamical Systems*, 27(1), 162–178. <https://doi.org/10.1080/13873954.2021.1889609>
- Jiménez-Gutiérrez, A. L., Mota-Hernández, C. I., Mezura-Montes, E., & Alvarado-Corona, R. (2024). Application of the performance of machine learning techniques as support in the prediction of school dropout. *Sci. Rep.*, 14(1).
- Kang, Z., Catal, C., & Tekinerdogan, B. (2020). Machine learning applications in production lines: A systematic literature review. *Computers & Industrial Engineering*, 149, 106773. <https://doi.org/10.1016/j.cie.2020.106773>
- Kritzinger, W., Karner, M., Traar, G., Henjes, J., & Sihn, W. (2018). Digital twin in manufacturing: A categorical literature review and classification [16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018]. *IFAC-PapersOnLine*, 51(11), 1016–1022. <https://doi.org/10.1016/j.ifacol.2018.08.474>
- Lai, T., & Robbins, H. (1985). Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1), 4–22. [https://doi.org/10.1016/0196-8858\(85\)90002-8](https://doi.org/10.1016/0196-8858(85)90002-8)

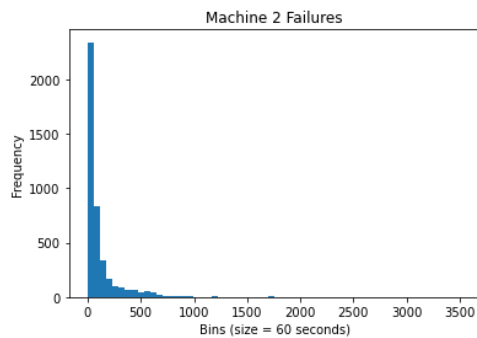
- Langford, J., & Zhang, T. The epoch-greedy algorithm for contextual multi-armed bandits [Cited by: 63]. In: Cited by: 63. 2008. <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85162018594&partnerID=40&md5=98a45bebd8975e929993a02a5087db06>
- Letard, A., Gutowski, N., Camp, O., & Amghar, T. (2024). Bandit algorithms: A comprehensive review and their dynamic selection from a portfolio for multicriteria top-k recommendation [Cited by: 1]. *Expert Systems with Applications*, 246. <https://doi.org/10.1016/j.eswa.2024.123151>
- M., A. (2014, January). *Simulation modeling and analysis* (5th ed.). McGraw-Hill Professional.
- Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Jia, Y., Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, ... Xiaoqiang Zheng. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems [Software available from tensorflow.org]. <https://www.tensorflow.org/>
- Monostori, L., & Prohaszka, J. (1993). A step towards intelligent manufacturing: Modelling and monitoring of manufacturing processes through artificial neural networks. *CIRP Annals*, 42(1), 485–488. [https://doi.org/10.1016/S0007-8506\(07\)62491-3](https://doi.org/10.1016/S0007-8506(07)62491-3)
- Nazabadi, M. R., Najafi, S. E., Mohaghar, A., & Sobhani, F. M. (2024). Agent-based decision making and control of manufacturing system considering the joint production, maintenance, and quality by reinforcement learning [Cited by: 0; All Open Access, Gold Open Access]. *Decision Making: Applications in Management and Engineering*, 7(1), 396–419. <https://doi.org/10.31181/dmame712024885>
- Olcott, S., & Mullen, C. (2020). *Digital twin consortium defines digital twin*. www.digitaltwin-consortium.org/2020/12/digital-twin-consortium-defines-digital-twin/ accessed 15-04-2024.
- Onehotencoder. (n.d.). Retrieved August 23, 2024, from <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html>
- Oriti, D., Brizzi, P., Giacalone, G., Manuri, F., Sanna, A., & Ordoñez, O. T. (2022). Machine learning and digital twin for production line simulation: A real use case [Cited by: 0]. *Lecture Notes in Networks and Systems*, 319, 814–821. https://doi.org/10.1007/978-3-030-85540-6_103
- Overbeck, L., Hugues, A., May, M. C., Kuhnle, A., & Lanza, G. Reinforcement learning based production control of semi-automated manufacturing systems [Cited by: 13; All Open Access, Gold Open Access, Green Open Access]. In: 103. Cited by: 13; All Open Access, Gold Open Access, Green Open Access. 2021, 170–175. <https://doi.org/10.1016/j.procir.2021.10.027>
- Panzer, M., Bender, B., & Gronau, N. (2022). Neural agent-based production planning and control: An architectural review. *Journal of Manufacturing Systems*, 65, 743–766. <https://doi.org/https://doi.org/10.1016/j.jmsy.2022.10.019>
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.
- Petsagkourakis, P., Sandoval, I., Bradford, E., Zhang, D., & del Rio-Chanona, E. (2020). Constrained reinforcement learning for dynamic optimization under uncertainty [21st IFAC World Congress]. *IFAC-PapersOnLine*, 53(2), 11264–11270. <https://doi.org/10.1016/j.ifacol.2020.12.361>
- Powell, W. B., & Frazier, P. (2008, September). Optimal learning. In *State-of-the-Art Decision-Making tools in the Information-Intensive age* (pp. 213–246). INFORMS.
- Shen, H., Hong, L. J., & Zhang, X. (2021). Ranking and selection with covariates for personalized decision making. *INFORMS J. Comput.*, (ijoc.2020.1009).

- Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.
- Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4), 285–294. Retrieved June 14, 2024, from <http://www.jstor.org/stable/2332286>
- Tremblay, F.-A., Durand, A., Morin, M., Marier, P., & Gaudreault, J. (2024). Deep reinforcement learning for continuous wood drying production line control. *Computers in Industry*, 154, 104036. <https://doi.org/10.1016/j.compind.2023.104036>
- Unpingco, J. (2019, July). *Python for probability, statistics, and machine learning* (2nd ed.). Springer Nature.
- Wu, D., Wang, Y., & Zhou, E. (2024). Data-driven ranking and selection under input uncertainty [Cited by: 2; All Open Access, Green Open Access]. *Operations Research*, 72(2), 781–795. <https://doi.org/10.1287/opre.2022.2375>
- Wuest, T., Weimer, D., Irgens, C., & Thoben, K.-D. (2016). Machine learning in manufacturing: Advantages, challenges, and applications. *Production & Manufacturing Research*, 4(1), 23–45. <https://doi.org/10.1080/21693277.2016.1192517>
- Xiao, H., Cao, M., Zhen, L., & Wang, X. (2024). Optimal computing budget allocation for selecting the optimal subset of multi-objective simulation optimization problems [Cited by: 0]. *Automatica*, 169. <https://doi.org/10.1016/j.automatica.2024.111829>
- Zhang, Z., & Jung, C. (2021). Gbdt-mo: Gradient-boosted decision trees for multiple outputs. *IEEE Transactions on Neural Networks and Learning Systems*, 32(7), 3156–3167. <https://doi.org/10.1109/TNNLS.2020.3009776>

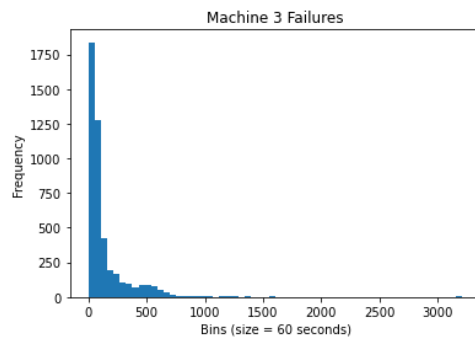
Appendix A

Machine failures

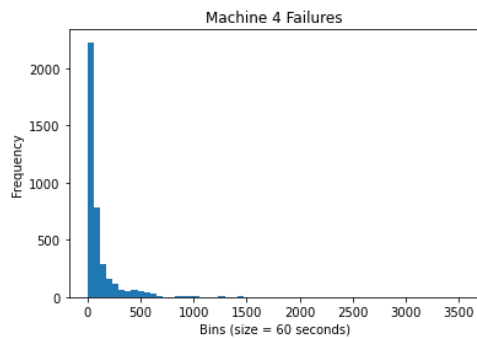
This appendix shows the graphs of the machine failures of machines two, three, four, nine and six.



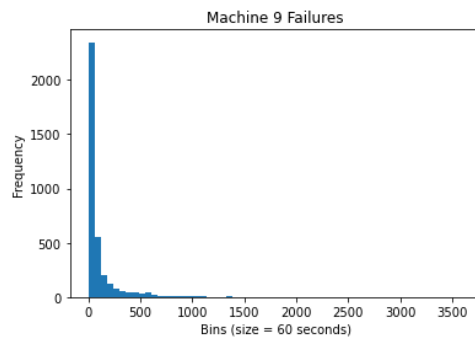
(a) Machine 2 Failures



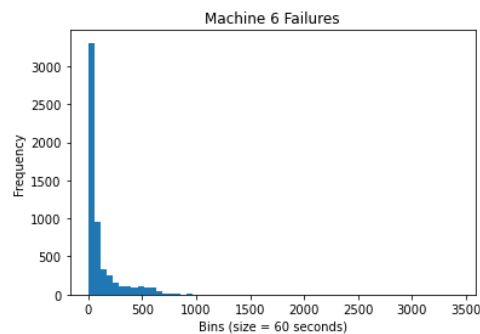
(b) Machine 3 Failures



(c) Machine 4 Failures



(d) Machine 9 Failures



(e) Machine 6 Failures

Figure A.1: Machine failures

Appendix B

Technical simulation description

This appendix describes the technical aspects of the simulation model, used to create a database, consisting of all data that is necessary to create a machine learning model. This machine-learning model should eventually predict the output that is given by the simulation model.

The simulation model provides output regarding given input. It evaluates what will happen in a coming time period, when the input is the starting state of the production line. It provides output in terms of number of items produced and wasted. Following an objective function, the simulation provides the expected output, given some random events. This means, that each time the simulation model is repeated, and the random number stream within the program is changed, then the output of the simulation model is different.

The next sections describe the process of the simulation model. First, the initialization, then the process of products along the simulation model and at last the calculation of several statistics.

B.1 Initialization

Before the start of a run, the real time data is initialized. This is done based on a table, RealTimeData, which consists of all variables that determine the current state of the line. These variables are the current line speed, buffer level and scenario that is used. But also the state of machines, the duration of failure, current item and its goal towards the full production of this item, and the item that is produced after this goal is reached. The simulation model initializes using these data, and continues its process from this items. This means that the conveyor belt is also filled with products. There is no warm-up period taken into account.

B.2 Process

The simulation model starts with products entering the factory through processing. Processing pushes products through to the packaging part of the production line, after which the products enter the line. At each entry to a machine, a check is build. This checks whether a row of products can enter to the machine. If this is possible, the row goes through two wrapping stations, before leaving the model. If this is not possible, the product remains on the conveyor belt.

When a product arrives at the last machine, there is first a buffer. The check that is built here, first checks whether there is room left in the buffer. If that is the case, the buffer will forward a product to the machine and takes a new row. If it is not possible to forward a row of products to the machine, the buffer will fill further.

If there is no space left in the buffer, the products will remain on the belt. These products will, at the end of the belt, leave the simulation model as waste products.

All products that are produced also leave the simulation model, but as produced, packed products.

The scenario, which is given as input, determines which machines can be switched off temporarily to accommodate faster production of other prioritized products. This is also checked each time a product enters the buffer. When the buffer levels exceeds the value given in Appendix C, a machine should be switched on again. However, when the buffer level decreases below a certain level, some scenarios enable machines to be switched off.

B.3 Statistics

At the end of a run, the simulation model calculates the amount of products produced and wasted. It also determines with this information the value of the objective function. We defined this objective function in Chapter 5. This objective value is corresponding to the input variables, which are defined in RealTimeData. With this data, the machine learning model is retrained.

Appendix C

Subscenarios

Sub Scenario	Buffer LB	Buffer UB	mc6	MC5	mc4	mc3	mc2	mc1
Fill Buffer	0	0.05	on	off	on	on	on	on
Norm.	0.05	0.1	on	off	on	on	on	on
Empty Buffer	0.1	0.4	on	off	on	on	on	on
Empty Max	0.4	1	on	off	on	on	on	on

Table C.1: Scenario 1

Sub Scenario	Buffer LB	Buffer UB	mc6	MC5	mc4	mc3	mc2	mc1
Fill Buffer	0	0.2	on	on	on	standby	on	on
Norm.	0.2	0.3	on	on	on	standby	on	on
Empty Buffer	0.4	0.5	on	on	on	on	on	on
Empty Max	0.6	1	on	on	on	on	on	on

Table C.2: Scenario 2

Sub Scenario	Buffer LB	Buffer UB	mc6	MC5	mc4	mc3	mc2	mc1
Fill Buffer	0	0.2	on	off	on	standby	on	on
Norm.	0.2	0.3	on	off	on	standby	on	on
Empty Buffer	0.4	0.5	on	off	on	on	on	on
Empty Max	0.6	1	on	off	on	on	on	on

Table C.3: Scenario 3

Sub Scenario	Buffer LB	Buffer UB	mc6	MC5	mc4	mc3	mc2	mc1
Fill Buffer	0	0.2	on	on	on	on	standby	on
Norm.	0.2	0.3	on	on	on	on	on	on
Empty Buffer	0.4	0.5	on	on	on	on	on	on
Empty Max	0.6	1	on	on	on	on	on	on

Table C.4: Scenario 4

Sub Scenario	Buffer LB	Buffer UB	mc6	MC5	mc4	mc3	mc2	mc1
Fill Buffer	0	0.2	on	on	on	standby	on	standby
Norm.	0.2	0.3	on	on	on	standby	on	on
Empty Buffer	0.4	0.5	on	on	on	on	on	on
Empty Max	0.6	1	on	on	on	on	on	on

Table C.5: Scenario 5

Sub Scenario	Buffer LB	Buffer UB	mc6	MC5	mc4	mc3	mc2	mc1
Fill Buffer	0	0.2	on	on	on	standby	on	on
Norm.	0.2	0.3	on	on	on	on	on	on
Empty Buffer	0.4	0.5	on	on	on	on	on	on
Empty Max	0.6	1	on	on	on	on	on	on

Table C.6: Scenario 6

Sub Scenario	Buffer LB	Buffer UB	mc6	MC5	mc4	mc3	mc2	mc1
Fill Buffer	0	0.05	on	on	on	on	on	on
Norm.	0.05	0.2	on	on	on	on	on	on
Empty Buffer	0.2	0.3	on	on	on	on	on	on
Empty Max	0.5	1	on	on	on	on	on	on

Table C.7: Scenario 7

Sub Scenario	Buffer LB	Buffer UB	mc6	MC5	mc4	mc3	mc2	mc1
Fill Buffer	0	0.2	on	on	on	on	on	standby
Norm.	0.2	0.3	on	on	on	on	on	on
Empty Buffer	0.4	0.5	on	on	on	on	on	on
Empty Max	0.6	1	on	on	on	on	on	on

Table C.8: Scenario 8

Sub Scenario	Buffer LB	Buffer UB	mc6	MC5	mc4	mc3	mc2	mc1
Fill Buffer	0	0.05	on	on	on	on	standby	on
Norm.	0.05	0.15	on	on	on	on	standby	on
Empty Buffer	0.4	0.5	on	on	on	on	on	on
Empty Max	0.5	1	on	on	on	on	on	on

Table C.9: Scenario 9

Appendix D

Simulation results

Speed	LB	Mean	UB

Table D.1: Sim results Validation sample 1

Speed	LB	Mean	UB

Table D.2: Sim results Validation sample 2

Machine	TimeToTarget	fraction
MC1	11:47:04.2424	0.09
MC2	2:31:30.9091	0.02
MC3	16:50:06.0606	0.14
MC4	1:22:17:46.6667	0.37
MC5	23:34:08.4848	0.19
MC6	1:01:15:09.0909	0.20

Table D.3: Priority + time to target calculations

Bufferlevel	mean
10	
20	
30	
40	
50	
60	
70	
80	
90	
100	

Table D.4: Sim results Validation sample no-linesettings

MC1	MC2	MC3	MC4	MC5	MC6	Speed
0	0	0	0	0	0	
900	0	0	0	0	0	
900	900	0	0	0	0	
900	900	900	0	0	0	
900	900	900	900	0	0	
900	900	900	900	900	0	
900	900	900	900	900	900	
1800	0	0	0	0	0	
1800	1800	0	0	0	0	
1800	1800	1800	0	0	0	
1800	1800	1800	1800	0	0	
1800	1800	1800	1800	1800	0	
1800	1800	1800	1800	1800	1800	
3600	0	0	0	0	0	
3600	3600	0	0	0	0	
3600	3600	3600	0	0	0	
3600	3600	3600	3600	0	0	
3600	3600	3600	3600	3600	0	
3600	3600	3600	3600	3600	3600	

Table D.5: Machine failures experiments