CHALMERS
UNIVERSITY OF TECHNOLOGY

Data-driven and Variant-based Throughput and Bottleneck Prediction Using Ensembled Machine Learning Algorithms

Master of Science Thesis in Production Engineering

ANTON JOHANNESSON
PERHAM SHAMS

Department of Industrial and Materials Science
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2018

Master's thesis 2018

Data-driven and Variant-based Throughput and Bottleneck Prediction Using Ensembled Machine Learning Algorithms

ANTON JOHANNESSON

PERHAM SHAMS

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Industrial and Materials Science
Division of Production Systems
Chalmers University of Technology
Gothenburg, Sweden 2018

Data-driven and Variant-based Throughput and Bottleneck Prediction Using Ensembled Machine Learning Algorithms
ANTON JOHANNESSON
PERHAM SHAMS

Supervisors:
Mukund Subramaniyan, Department of Industrial and Materials Science, Chalmers
William Falkenström, Virtual Manufacturing
Erik Halvordsson, PDSVision
Examiner: Anders Skoogh, Department of Industrial and Materials Science, Chalmers

Master's Thesis 2018
Department of Industrial and Materials Science
Division of Production Systems
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2017

Data-driven and Variant-based Throughput and Bottleneck Prediction Using Ensembled Machine Learning Algorithms
ANTON JOHANNESSON
PERHAM SHAMS
Department of Industrial and Materials Science
Chalmers University of Technology

Abstract

The manufacturing domain is currently shifting to a new era, Industry 4.0. The era relies heavily on big data and digitalisation which puts large pressure on the development of reliable tools to analyse the data. The recent advances in research within machine learning allows for quick analysis of large amounts of data to shine light on important information and patterns. Presently, machine learning in manufacturing is mainly used for single station applications, but the technology is now mature enough to be applied to the manufacturing domain in a more holistic, system-wide manner.

The thesis aimed to evaluate how the throughput and bottlenecks could be predicted for an entire production line by applying machine learning, and how much data was required for the training of such a model. The Internet of Things platform ThingWorx, which has an integrated machine learning platform was the software utilised to create a pattern recognition algorithm. This was done by using the order sequence as input to a machine learning model which was trained on data from a discrete event simulation.

This resulted in a successful machine learning model for predicting throughput, while the bottleneck prediction could not reach satisfactory levels of accuracy. The throughput prediction could reach 97.8% accuracy with data from one year of production. The possibility of predicting if a specific production hour is going to produce a desired amount of products can help production leaders in crucial decision-making and proactive actions can be performed.

Keywords: Machine learning, bottleneck prediction, throughput prediction, big data, ThingWorx, Industry 4.0, Internet of Things.

Acknowledgements

This master thesis has been written at the production engineering program at Chalmers University of Technology and in cooperation with Virtual Manufacturing, PDSVision and Adient in the spring of 2018.

We would like to thank Mukund Subramaniyan, at the Department of Industrial and Materials Science at Chalmers Technical University, for his help and insight in both statistical analysis and machine learning.

We also want to give recognition and a thank you to Erik Halfvardsson from PDSVision for making this thesis possible to begin with, and for his enthusiasm in pushing IoT, Big Data and Machine Learning into the industry.

We would also like to thank William Falkenström from Virtual Manufacturing for his great and swift help concerning the DES modelling required to perform this project.

We would like to express our gratitude towards Arvid Nilsson from Virtual Manufacturing, and Adient at large, for the possibility to base this case on Adient's production line and their willingness to share the challenges and opportunities that are present in today's industry.

Last but not least, we would like to thank Virtual Manufacturing at large for hosting us. Their openness toward new technologies and insight in what is value-adding has guided us toward a target which we are very satisfied with and which we believe will be applicable in the future.

Anton Johannesson and Perham Shams, Gothenburg, June 2018

Contents

List of Figures

List of Tables

1
Introduction

In this chapter the background of the project is presented, followed by the gap in the research domain that this thesis will attempt to fill. Further, the project purpose and aim is defined, and lastly the scope and delimitations are described.

1.1 Background

The domain of manufacturing is currently in an era of digitalisation (Zhou, 2013). There is a lot of buzz regarding Industry 4.0 (Lasi et al., 2014), Internet of Thing (Bi et al., 2014) and Big Data (Yin and Kaynak, 2015). These areas all strive to monitor, save and distribute data from different parts of an enterprise to enable a faster, more informed decision making process throughout a company. With the new paradigm of mass customisation, even mass personalisation, the pressure on the manufacturers to be flexible and fast is larger than ever. Short lead time and time-to-market are key concepts for staying ahead of the curve and digitalisation is the way to do it (Jeschke et al., 2016).

Coinciding with the great opportunities of Big Data are the large challenges (Yin and Kaynak, 2015). The amount of data that is being collected can overwhelm the personnel responsible for utilising it and the data can often remain unused (Tole, 2013). This results in investment costs of implementing the data collection as well as costs for maintaining and storing it, without any added business value.

To solve the challenges of Big Data and take advantage of the opportunities, machine learning can be utilised. Machine learning is a very mature technology and there are many domains where it has been adding value for decades (Konenenko, 2001). However, there is a lot of untapped potential in the manufacturing domain when it comes to machine learning, since most of the focus has been on machine level rather than system level. Manufacturing is a very complex domain where the interdependence between different parts is very subtle and can be far too complex for a human to analyse with many moving parts. Thus, this thesis aims to fill a niche of the gap regarding utilisation of machine learning to analyse the manufacturing domain on a more holistic level, focusing on how the variant mix differs in a short-term perspective and how this affects key performance indicators such as throughput per hour and bottlenecks.

1.2 Gap

Machine Learning is an area that currently is being researched heavily in several different domains. Ranging from customer experience (Yan et al., 2001) to autonomous cars (Kuderer et al., 2015). No matter what area, machine learning is used to analyse large amounts of data and improve decision making. This can speed up tasks that require analysing skills and can, in some cases, achieve results that are far better than the human mind is capable of (Ciresan et al., 2012).

When it comes to the manufacturing domain there has also been an interest in machine learning. The usage often leans toward predictive maintenance (Susto et al., 2015), which aims to predict when a machine is close to a failure. This would allow maintenance to be performed prior to the breakdown, but without the loss from performing maintenance too often, which can be a drawback from preventative maintenance. However, this is a fairly narrow niche within the area of manufacturing since it is in most instances focusing on one machine at a time.

Although there has been a fair amount of research performed in regards to machine learning on a system level within the field of manufacturing, the focus has often been towards how breakdowns and quality parameters affect bottlenecks and throughput (Cao et al., 2012). When breakdowns are included the time span of the prediction is often fairly large, at the very least longer than the time between breakdowns.

With an increase in Assembly-To-Order (ATO) based production and deliveries by a strict Just-In-Time (JIT) methodology, the variant mix can no longer be planned and the variant distribution can vary heavily on a short-term basis. Considering how balancing in a production system is often based on a specific distribution of variants, the balance can be very poor for specific hours. With this in mind, it is now interesting to evaluate how the throughput and bottlenecks of a production system can be predicted with machine learning depending solely on the variant sequence for the set time interval.

1.3 Problem statement

A demand for short lead times and a requirement for high flexibility are put on Adient from their sole customer. Since the product orders includes several different variants to produce, it puts a lot of pressure on the production system to perform efficiently and without issues to provide quality products in time.

The importance of making sure the production supervisors are able to find problems on the line quickly and have adequate information about the incoming order sequence to counter future potential problems is therefore evident. This is to ensure a reliable and stable production system with minimal delays and reduced overtime. The number of possible combinations due to the substantial amount of variants in the order sequence is immense and makes it far too complex for a human to analyse.

1.4 Purpose

The purpose of this thesis is to generate information which can increase productivity by facilitating data-based decisions in production. Thus, the capabilities of machine learning will be evaluated on a system level of the production line to attempt to access previously unknown information.

1.5 Aim

The aim of the thesis is to build a functioning machine learning model which has the capabilities to predict how the variant sequence will affect the throughput during a set time frame. Furthermore, the model aims to predict the station on the line where the bottleneck is situated. By identifying and informing the production supervisors prior to possible issues, measures can be taken which allows for an overall more balanced production and an enhanced production planning. The result has the possibility to lead to better planned overtime, preparation for delays and increased productivity.

In connection to the machine learning model, the intention of the thesis is to investigate and evaluate what sort of production data that is required to create such a model and how to, in an optimal way, establish a link between the order sequence and the output of the model. Finally, the thesis aims to evaluate the method of using the software ThingWorx Analytics for modelling of a machine learning model by investigating its advantages, specifically for Adient's production line scenario.

1.6 Scope and delimitation

The scope of this thesis does not include the development of a specific machine algorithm adapted for the case, but it will include the utilisation of existing algorithms to create and build the machine learning model. The thesis will utilise the IoT platform ThingWorx, which incorporates six different algorithms that can be applied. Therefore, the thesis is limited to the use and combination of these six models available in ThingWorx Analytics, which are linear regression, decision tree, neural networks, random forest, gradient boosting machine and logistic regression.

The scope for the machine learning model involves identifying if there is going to be a problem, i.e. a production goal not being reached, or where the problem lays, i.e. location of the bottleneck. The scopes does not include how to solve the potential problems found, for instance, improvement potentials for the affected station or how to rearrange the order sequence.

In order for the project to fit in the time frame for a master's thesis the data sampling is relying solely on Adient's ability to provide historical data. No new sensors are mounted on the actual stations and no time studies are to be conducted. Additional data that could potentially be needed, e.g. for bottleneck detection, is generated through a discrete event simulation (DES) model which is based on process times, down-times, setup times and the production flow logic from the actual production line.

1.7 Product and production system of the case company

The products and production system that is being investigated is owned by a company named Adient, a world leader within seating solutions in the automotive industry. The thesis is focusing on their plant in Torslanda, Sweden, where seats for Volvo Cars are produced in close proximity to the Volvo Cars plant.

1.7.1 Production strategy

As the paradigm within production currently lies within the mass customisation era, and is shifting into the even more demanding paradigm of mass personalisation (Jack Hu, 2013), manufacturing companies have to deal with large amounts of variants. In order to be able to satisfy the demand for the large amount of variants, it is now common to adopt a pull-based production system within the automotive industry (Bhamu and Sangwan, 2014), which is also true for Adient. Adient's plant in Torslanda is producing in a pull-based manner, i.e. ATO, strictly following the order sequence of its only customer, Volvo Cars. Furthermore, Volvo Cars is relying on JIT deliveries in order to keep inventory low. JIT is a fairly old method to reduce the need for a vast storage, resulting in lower bound capital (Frazier et al., 1988).

With a pull-based production system in combination with JIT deliveries, and with a growing amount of variants, the balancing of the production line is a complex task. Allowing customers to add extra features to the seats, which requires extra tasks in the production, is hypothesised to lead to variation in regards of bottlenecks and throughput for the production line.

1.7.2 The production line

The production line being investigated is one of Adient's rear seat production lines, named V54x, which is one of four production lines presented in figure 1.1. V54x is a straight production line that contains 47 stations and these stations are called RSA-01, RSA-02 etc. up to RSA-47. 27 out of these stations are manual assembly stations and are manned by one operator individually. The remaining stations are either machines or buffers with the capacity of one. The machines on the line consists of an infrared oven used as a heating process for the seats and a headrest pull tester that uses a specific force to test if the headrest is locked into place. Beyond these 47 stations, there exists a rework loop that works as a larger buffer and brings defect products to a station that has an operator that adjusts the product. The product is then reintroduced to the line at a merging point.

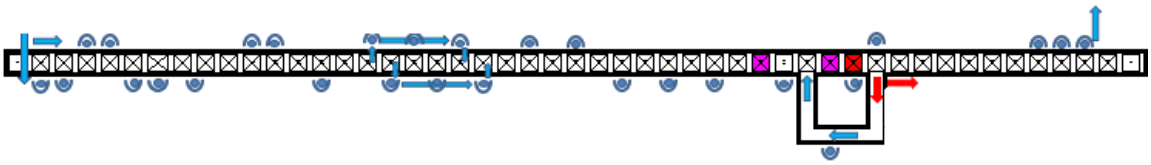

Figure 1.1: A layout displaying the main section of the rear seat production line, V54x, at Adient.

Currently, V54x has a planned takt of 44 rear seats per hour. However, due to circumstances related to uneven balancing and short stoppages, the planned takt can be hard to reach. The line is running for 24 hours a day, with three consecutive shifts, Monday to Friday. Overtime is decided at 9PM on Friday at the latest and takes place on Saturday. At the moment, overtime on Saturdays occurs every week because of the takt being lower than demanded. A preliminary order sequence is set 10 weeks before the planned delivery and is finally locked in two to four hours before delivery depending on how much the production is behind on schedule.

1.7.3 The product

The product manufactured at the V54x line is the so called rear seat, which can be described as the connected passenger seats directly placed behind the driver and front passenger seat in a standard automotive vehicle. The rear seat is produced in two different parts throughout the production line, which are then assembled at the merge point next to the end station. One of the parts is called the 60% version, and as the name suggests it is comprised of 60 percent of the rear seat, i.e. the middle seat section in addition to one of the side seats. The 40% version is comprised of the remaining side seat. The parts that are pairs are manufactured alternatively with the same complements.

The rear seat to three different Volvo car models are manufactured at the production line, and these can be seen in table 1.1. Additional features can be added by the customer, which results in the production system having to be able to manufacture unique products in accordance to customer demand and therefore is required to handle several different variants. In table 1.1, it is possible to observe the number of variants that differ in process time for each seat and car model. However, there are many features that do not affect the process time, but differ in appearance which makes it impractical to have a storage of finished products.

Table 1.1: Car model, seat model and number of variants produced on the V54x production line at Adient.

Car model (Volvo)	Seat model (Adient)	Amount of Variants
V90	RS V542	8
V90 CC	RS V543	8
XC90	RS V426	8

2
Literature review

In this chapter, an overview is given on the related theory and research for this thesis. It serves the purpose of giving the reader a sufficient understanding about why and how the project is performed and also to provide important knowledge in the domain of machine learning.

2.1 Frame of reference

This section covers the frame of reference of the project. The frame of reference is necessary to fully understand the current state of research within the area of machine learning applied in the manufacturing domain, but also supporting information that is required - such as bottleneck detection.

2.1.1 Bottleneck detection

In a serial production line it is sure to be at least one bottleneck (Goldratt, 1990). A bottleneck is the machine or station that constrains the throughput of products produced in the production system (Betterton and Silver, 1988). Constraining the system means that it causes blockage upstream and starvation downstream, limiting how much the rest of the system can produce. This means that improvements made at any other location than the bottleneck will yield low results in term of throughput. Thus, it is crucial to identify the bottleneck to efficiently allocate resources to the machine which will elevate the production system the most (Goldratt, 1990).

The need for bottleneck identification has been known for a long time and a lot of research has been put into formulating different types of approaches to accurately find the bottleneck (Betterton and Silver, 1988). The effectiveness of each approach often depends on how fitting it is to the specific production system (Subramaniyan et al., 2016b). This in turn means that a thorough analysis of the production system has to be conducted in order to choose a fitting bottleneck identification approach. Moreover, the need for continuous monitoring of the bottlenecks is required since the constraint of the system can relocate during the production run (Subramaniyan et al., 2016b).

With a holistic view the bottleneck identification methods have often been divided into three sub-categories, namely analytical, simulation based, and data-driven (Subramaniyan et al., 2016b). Many of the methods have been updated and improved since they were first implemented. An example of this is shifting bottleneck by Roser et al. (2002) that has been improved to a data driven method by Subramaniyan et al. (2016b) and will be listed with the latter as the reference in the table, see table 2.1.

Table 2.1: Listing the most common bottleneck detection methods, adapted from Subramaniyan et al. (2016b) but updated with recent advancements in the data-driven sector.

Interpretation	**Method**	**Reference**
Analytical	Queue Length	Lawrence and Buss (1994)
Analytical	Utilization	Hopp and Spearman (2001)
Simulation/model based	Active period percentage	Roser et al. (2001)
Simulation/model based	Queue time	Faget et al. (2005)
Simulation/model based	Inactive Period	Sengupta et al. (2008)
Simulation/model based	Inter-departure time variance	Betterton and Silver (1988)
Simulation model coupled with real-time input data	Sensitivity based bottleneck detection	Chang et al. (2007)
Data-driven	Turning point	Li et al. (2009)
Data-driven	Shifting bottleneck detection	Subramaniyan et al. (2016b)
Data-driven	Average active period	Subramaniyan et al. (2016a)

As can be seen in table 2.1 most of the methods are based on either simulation or data-driven, meaning that they have a descriptive nature. According to Subramaniyan (2015) up to 100 rows of data is stored per machine and hour, leading to the assumption that large parts of the industry already have mature Manufacturing Execution Systems (MES) that are able to handle the required type of measurements. This facilitates the possibility of data-driven algorithm for continuous bottleneck detection.

It is important to continuously monitor the bottleneck since it may vary throughout the production run (Subramaniyan et al., 2016b). The variations can be due to different parameters such as down times, setup times or variation in process time. The goal of bottleneck detection is to allocate resources in an optimised manner, and if the bottleneck shifts the resource allocation has to follow (Goldratt, 1990).

2.1.2 Bottleneck prediction in manufacturing

The importance of continuous bottleneck detection has been known for a few years, and the next step is to predict bottlenecks to facilitate proactive measures rather than reactive. In table 2.2, some unique examples of predictive methods are presented that were developed for predicting the future bottlenecks within a production system. Important to note is that the three different prediction methods presented below are basically the extent of the system level machine learning applications within manufacturing.

Cao et al. (2012) has performed a study where they utilize the Adaptive Network-based Fuzzy Inference System (ANFIS) algorithm specifically to predict bottlenecks in a semiconductor manufacturing system. A bottleneck index was derived and used to predict the bottlenecks of the system using variables such as processing times, utilisation rates, work-in-progress (WIP), buffer lengths, product variants etc. The output of the model was set to be the stations predicted to be the primary and secondary bottleneck. The result from the study was that the ANFIS algorithm achieved an accuracy of 92.03% for predicting the bottleneck for their specific system.

A different method for predicting bottlenecks was suggested by Li et al. (2011) which involved using the method of Auto-regressive Moving Average (ARMA) that utilises time series of data. The study was performed for a small automotive assembly line and the bottleneck detection was based on the turning point method (table 2.1), where the time series data collected was the sequence of observed blockage and starvation times. The ARMA model was used for its ability to indicate dependence in the time series, which lead to great performance when it came to predicting future values in the observed series, and in this case predicting blockage and starvation times. The ARMA model managed to predict the blockage and the starvation times with an accuracy of 97.38%.

The connection between the studies made by Cao et al. (2012) and Li et al. (2011) is that both utilise DES for their respective production system to acquire production data. This means that the accuracy for predicting bottlenecks are based on a simulation environment, instead of being based on real-time data. Subramaniyan et al. (2018) emphasises the importance of using real-time data to fully be able to claim the algorithm performance and to establish trust in the algorithm. To address the lack of real-world validation for bottleneck prediction algorithms, Subramaniyan et al. (2018) performed a study utilising real-time MES data for developing a bottleneck prediction algorithm using Auto-Regressive Integrated Moving Average (ARIMA). The established algorithm was data-driven and used the active periods of the machines as a technique to find the bottlenecks in the system. The algorithm from the study was implemented and tested over a real-world automotive production line and managed to achieve an accuracy of 86.13% and recall of 62.53% for predicting a group of bottlenecks. Subramaniyan et al. (2018) proposed the employment of using a standard predictive algorithm, the naïve method, to make an evaluation of the performance of the developed algorithm for the system.

Table 2.2: Overview of previous work related to bottleneck prediction in manufacturing.

Prediction Method	Input	Output	Results
ANFIS, Cao et al. (2012)	MTTR, MTBF, Process time, WIP, Buffer lengths, Utilisation rates	Primary bottleneck: Secondary bottleneck	Accuracy 92.03%
ARMA, Li et al. (2011)	Blockage period, Starvation period	Forecasted blockage, Forecasted starvation	Accuracy 97.38%
ARIMA, Subramaniyan et al. (2018)	Active periods: Processing, Blockage, Starvation, Idle, Down	Forecasted active period, Associated standard error	Accuracy of 86.13%, Precision of 36.34%, Recall of 62.53%

2.1.3 Throughput prediction in manufacturing

In table 2.3, three unique predictive methods are presented that were developed for predicting the throughput within a production system. In this section those three methods are described more closely.

Pandian and Ali (2013) have performed a study on an automotive assembly line where they utilise both ARMA and ANN algorithms separately to predict the production loss in the system. To develop the prediction algorithm real time data was used with variables such as machine downtime, blockages, production stops and machines with slow process times. The result from the study can be observed in table 2.3 and the better predicting algorithm depended on what line that was being analysed. However, both gave similar results.

In a study made by Samattapapong and Afzulpurkar (2016) the predictive system was more focused on forecasting the throughput for an automated hard disk drive test operation. A type of ANN was used called Generalised Regression Neural Network (GRNN), and the input was created from historical data (time stamps and product volumes in and out of processes) from the manufacturing processes. This resulted in a forecasting algorithm, that was able to provide predictions with a MAPE of 1.1%.

Azizi (2016) conducted a study combining the prediction methods of Bayesian and ARIMA to try to predict the production throughput for production systems which are complex due to random variables. The case that was studied involved a real production of tile and ceramic. Random variables from the production such as demand, breakdown time, scrap, set-up time, and lead time, were used to develop the algorithm. The result from the study indicated that by combining the two methods Azizi (2016) was able to increase the predication accuracy compared to using each method individually and the model resulted in a R-squared value of 98.8%.

Table 2.3: Overview of previous work related and relevant to throughput prediction in manufacturing.

Prediction Method	**Input**	**Output**	**Results**
ANN & ARMA, Pandian and Ali (2013)	Machine downtime, Blockages, Production stops, Slow cycle times	Production loss	ANN (MAPE): 0.35%, 0.30% ARMA (MAPE): 0.20%, 0.34%
GRNN, Samattapapong and Afzulpurkar (2016)	I/O times, I/O product volumes	Prod. throughput	1.1% MAPE
Bayesian-ARIMA, Azizi (2016)	Demand, Breakdown time, Scrap rate, Set-up time, Lead time	Prod. throughput	R-squared: 0.988

2.1.4 Conclusions from frame of reference

There are some conclusions that could be drawn from analysing the related research. ANN, ARMA and ARIMA algorithms (see table 2.2, 2.3) seems to be the most frequently chosen and tested methods for predicting both bottlenecks and throughput in the domain of manufacturing. The research suggests that they are very successful at reaching high prediction accuracy for a variation of input parameters from different areas and processes in manufacturing. ANN is an integrated method in ThingWorx Analytics and may therefore play an important part in the project. Moreover, the studies connected to bottleneck prediction points at the importance of choosing the right bottleneck detection method for the specific case when creating a prediction model to ensure high accuracy.

When analysing the different predictive algorithms presented in table 2.2 and 2.3, it is evident that most of the methods only utilise machine data for their input and many focus on trying to predict the random behaviours of the machines using variables such as repair rates, quality, down-times, stops etc. What is interesting is that very few of these models take the product categories into account. Furthermore, many studies have been performed on experimental and small scale production or on machine-level and not a lot of research have been made taking a more holistic view, looking at a larger production line. Many of the cases look at highly automated production systems while there is a lack of research when it comes to including manual work stations in the analysed system. Moreover, there is a lack of information on the quantity of historical data needed to optimise certain algorithms and how the size of the data will affect the accuracy and error for the algorithm. It is therefore clear that the research and results from this thesis can help reduce the aforementioned research gap and shed more light on data-driven, variant-based prediction methods for throughput and bottlenecks.

2.2 Machine learning terminology and tools for prediction

In this section, to create a better understanding, an introduction and description are presented over terminology and methods used in the project that are connected to machine learning.

2.2.1 Feature vectors and attributes, input space and feature space

In machine learning there is certain terminology that is important to be aware of. It is important to know the difference between input space and feature space. In the input space the raw data is stored. The simplest example is image recognition, where the images remain in the input space (Camps-Valls et al., 2007). However, a machine learning model requires input in the form of numbers or classes. In order to generate suitable input to a machine learning model features have to be extracted, in image recognition this could be colours, shapes or other types of image features. This leads to a feature vector for each individual image, where each position in the feature vector is called an attribute (Camps-Valls et al., 2007).

2.2.2 Unsupervised and supervised learning

There exists several different types of machine learning algorithms. One important division that is made between the types of algorithms are if they follow the principles of unsupervised learning or supervised learning. Unsupervised learning, often referred to as clustering, finds similarities in the the feature vector and groups them together (Längkvist et al., 2014).

For the supervised learning, the machine learning model is told the true output in the training set (Kotsiantis et al., 2006). This means that instead of finding similar feature vectors, it is told what output the feature vector represents and treats them as a group. These two areas of machine learning methods have two different key uses. Supervised learning is mainly used when the pattern is already known, while unsupervised learning is used to detect new, unknown patterns.

2.2.3 Continuous and categorical variables

In machine learning and statistical analysis it is very important to make the distinction between categorical and continuous variables. Categorical variables are those for which predictions from a model can be put into classes, often referred to as classifiers. There are according to Agresti (2014) three different ways of categorising variables. The simplest one is when there only exists two categories, such as "Yes" and "No", and those are called binary variables. Furthermore, the categorical variables can have an ordinal scale, for example classifying them as: very bad, bad, average, good and very good. These are called ordinal variables. Finally, there are the nominal variables. For these variables the order is not relevant since they are independent of each other. One example can be taken from the project case where the bottleneck stations are nominal variables (Station 1, Station 2, Station 3, etc.) In contrast, continuous variables are those for which predictions from a model can be an infinite number of possible values. An example from the project is the predictions made on the production throughput, where the model can predict a range of real numbers, i.e. continuous variables, as close to the goal as possible.

2.2.4 Regression and classification models

Predictive modeling can be divided into two different model types, classification and regression. The difference between these two model types are connected to continuous and categorical variables (section 2.2.3). The classification model is used when the target variables for the predictive model is a set of categorical variables. The regression model is instead used when the target prediction is a continuous quantity (Gareth et al., 2015).

2.2.5 Linear regression model

When the task at hand is to find the relationship between different variables which are continuous, one of the oldest techniques is to use regression analysis (Yan and Su, 2014). In machine learning, one of the more common methods of regression is linear regression, which utilises linear predictor functions to model the relationships between observations and predictions (Seal, 1967).

The method approach for linear regression is simple compared to many other machine learning algorithms. The input is a vector of features, x, which are multiplied with an individual weighting variable for each specific vector spot. The vector is then summed up to give the expected output. The weighting variable, w, is calculated by inserting the training set into the model which then finds the optimum weighting for each index in the feature vector. See equation 2.1.

$$\sum_{i=0}^{n} w_i x_i = w_0 x_0 + w_1 x_1 + w_2 x_2 + w_n x_n \tag{2.1}$$

2.2.6 Decision trees

Decision trees are one group of machine learning algorithms. The concept of the decision tree can be explained as a classifier that models several decisions which takes the form of a rooted tree. The decision tree structure (see figure 2.1) is built up by outgoing nodes called internal nodes which describes the condition, which then splits into branches leading to another condition or into a final decision based on the path that was taken throughout the tree (Maimon and Rokach, 2010).

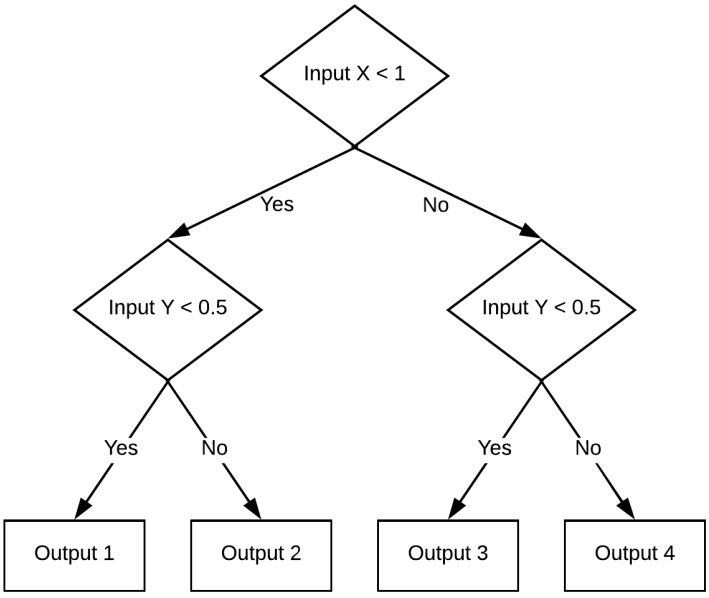

Figure 2.1: A simplistic classification decision tree structure comprised of internal decision nodes and predictions.

There are several general advantages and disadvantages using the decision tree method. The advantages are that decision trees are not very complicated to understand and it is easy to visualise the concept. Moreover, it has the capability to manage multiple outputs from the system. It is also said by Gupta (2017) that in the case of decision trees, minimal pre-processing and preparation of the data is required and that it can handle large data sets in a reasonable time. It is also possible to create both a classification or regression model with decision trees, depending on the decided goal variable (Gareth et al., 2015). For the disadvantages, there is a risk of overfitting (Dietterich, 1995) caused by the model creating over-complex trees. Furthermore, decision trees are also said to be unstable against minimal variations in the data (i.e. variance) and according to Gupta (2017) there exist methods such as bagging and boosting that can lower the variance if it is a problem, which means that several decision trees are used in combination.

2.2.7 Artificial neural network (ANN)

ANN is a machine learning method that is very popular and is being used increasingly within a vast range of fields, such as, finance, medicine (Dawson, 2016), manufacturing (Baseri and Belali-Owsia, 2017), vehicle control (Zissis et al., 2015) and many more. What makes ANN interesting to a lot of people is that the computational model is inspired by the biological neural networks from the brain of humans (Shanmuganathan and Samarasinghe, 2016). However, ANN has the capabilities to perform effective pattern recognition and numeric predictions that are too complicated for the human brain to handle (Dawson, 2016).

As with the likeness of a human brain, ANNs consists of processing elements called neurons that are interconnected and creates an artificial network (Shanmuganathan and Samarasinghe, 2016). These neurons have the capability to adapt to the circumstances, i.e. learn from experience (Dawson, 2016), and are doing so by changing their connections in the network (Gerven and Bohte, 2017). Each connection has an assigned weight (coefficient) that adapts throughout the learning process and helps strengthen or weaken the signal (Shanmuganathan and Samarasinghe, 2016).

The ANN structure can be divided into three different layers, which are input, hidden and output. The signals travel from the input layer, through a set number of hidden layers, where the value of the signal will transform and take different paths by each neuron it passes (recognising and training patterns), until it reaches the output layer with the prediction (Dawson, 2016). In figure 2.2, an example of what a simple ANN structure looks like can be observed, where the input layer will be representing and containing the order sequence information, and the output layer will be representing the predicted throughput.

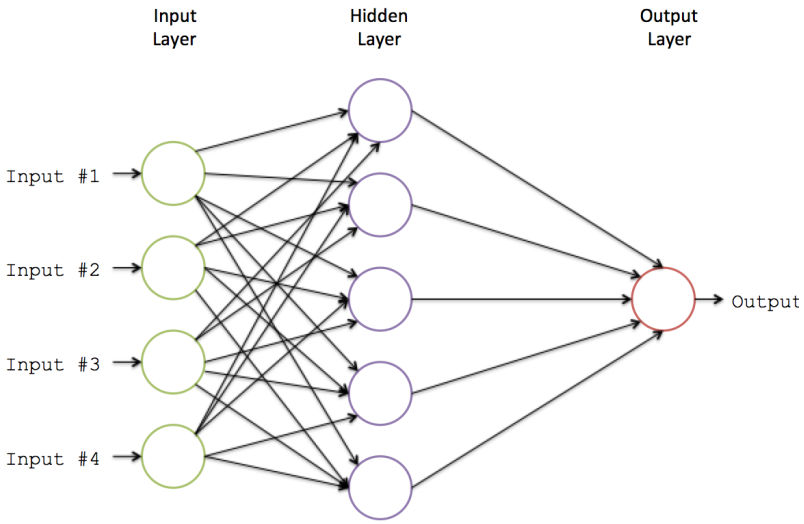

Figure 2.2: A simplistic ANN structure containing input, hidden and output layers.

2.2.8 Ensemble technique

Depending on what is being evaluated with machine learning the suitability of different machine learning techniques varies. It often depends on if the input is continuous or categorical, and if the output is continuous or categorical. If the input and output type is known there are still many techniques to choose between and the decision is rarely trivial. To reduce the complication of choosing a suitable machine learning algorithm, an ensemble model can be used (Opitz and Maclin, 1999).

Ensemble models trains several different machine learning algorithms using the same data set. During validation the accuracy of each individual machine learning algorithm is evaluated based on a statistical metric. The ensemble model then chooses the best composition of machine learning models to base the result on, which can range from one, to all of the trained machine learning algorithms (Opitz and Maclin, 1999).

2.3 Evaluation metrics for the prediction models

There exist numerous statistical tools that can be utilised to make an evaluation of the accuracy, or more widely used, the goodness of fit of different models. Accuracy is more of a binary way of evaluating a model since it is either correct or incorrect, and on the other hand goodness of fit focuses more on statistical metrics and is therefore more fluid. The methods usually involve comparing the generated values from a model, i.e. the predictions, with the actual values, called the observations, in different ways. Each method has its own advantages and drawbacks and deciding which one to use will heavily depend on the shape and size of the data. What is important and most desirable when choosing methods, is to look at their capabilities to discriminate among models, i.e. showing high variance between model results (Chai and Draxler, 2014).

2.3.1 Root-Mean-Square Error (RMSE)

RMSE is a very popular method for statistical modeling and is regularly used to compare the results from predictive models (Armstrong and Collopy, 1992). The concept of RMSE is fairly simple. By calculating the standard deviation of the prediction errors, the so-called residuals, RMSE will give an estimation of how well the data fits the regression line for the observed values (Barnston, 1992). The lower the resulting RMSE-value is, the better the chance that model has an exceeding goodness of fit compared to other models using same scale data.

The equation used for the RMSE-formula can be observed in equation 2.2, where P is the predicted values, O is the observed values and n is the amount of data points being compared.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(P_i - O_i)^2}{n}} \tag{2.2}$$

An important aspect and common concern when using RMSE as an evaluation tool is that it is more sensitive to outliers in the data compared to other methods, especially when the data samples are small (Barnston, 2006). If larger outliers do exist, it might be necessary to either filter them out before calculating RMSE (Chai and Draxler, 2014) or look to use methods using percentage error instead, which has the advantage of being less sensitive to outliers and not being dependent on the scale of the data to make comparisons.

2.3.2 Mean Absolute Percentage Error (MAPE)

MAPE is a common method used to measure prediction accuracy and as the name suggest it expresses the percentage of the mean prediction error. Thus, MAPE gives a better indication (compared to RMSE) of the size of the prediction errors relative to the size of the data points in the data set (Barnston, 2006). The calculation of MAPE can be observed in equation 2.3, where O is observed value, P is the predicted value and n is the number of predicted and observed data points. MAPE is widely used to compare different predictive models to each other.

$$MAPE = \frac{100}{n}\sum_{i=1}^{N} \mid \frac{O_i - P_i}{O_i} \mid \tag{2.3}$$

2.3.3 Sensitivity analysis: t-test

To be able to test the sensitivity of the results and establish if there is any statistical significance between two related observations, it is possible to utilise the paired t-test. The paired t-test takes the mean and standard deviation from two normally distributed observations and gives a t_{val} which indicates if there is a significant difference between the two observations. The equation for the t_{val} is presented in equation 2.4, where X and Y is the mean of the compared observations, and SE is the respective standard error for each set of observations. If the t_{val} is less than -1.96 or t_{val} is greater than 1.96 when using a confidence interval of 95%, then there exists a statistical difference between the two resulting observations (Subramaniyan et al., 2018).

$$t_{val} = \frac{(X - Y)}{\sqrt{SE_x^2 + SE_Y^2}} \tag{2.4}$$

2.3.4 Confusion matrix

When faced with statistical classification, i.e. if the predictive model is predicting classes, then it can be of interest to create and use a confusion matrix. A confusion matrix gives a clear indication of the model classifier performance and a good overview of the distribution of classes that are predicted well respectively poorly.

The confusion matrix is most easily explained by giving an example of a binary class problem and prediction. The matrix is in this case first assigned two sides, the observed classes (Yes and No) and the predicted classes (Positive and Negative). The matrix is then further divided into four different sections that depicts the different outcomes of the model. If the observed variable is positive and is classified as positive this will result in a true positive (TP); if it is classified as negative this will result in a false negative (FN). If the observed variable is negative and is classified as positive this will result in a false positive (FP); if it is classified as negative this will result in a true negative (TN) (Deng et al., 2016). See figure 2.3 for an illustration of the example.

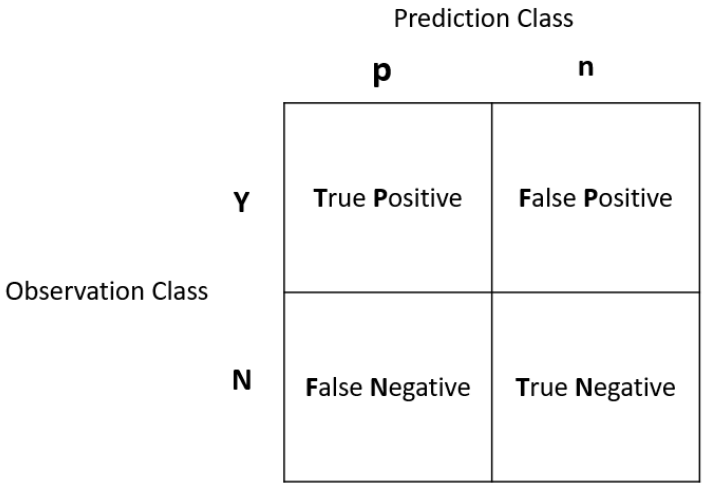

Figure 2.3: Simple structure of a confusion matrix.

$$Accuracy = \frac{TP + TN}{Total} \tag{2.5}$$

$$Precision = \frac{TP}{TP + FP} \tag{2.6}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.7}$$

From the confusion matrix, three different metrics can be calculated and used to evaluate the results. These derived metrics are accuracy, precision and recall and can be seen in equations 2.5-2.7. The accuracy will indicate how frequently the model predicts the classes correctly. The precision will give the fraction of the positive predictions that were correct. The recall will indicate the portion of the correct predictions that were true positive.

2.4 Benchmarking of the prediction model

In order to be able to evaluate whether or not a prediction model is actually adding value, benchmarking methods have to be applied. In this section the benchmarking methodologies will be explained. The method that has the highest accuracy for each objective will be used.

2.4.1 Naïve method

The naïve method is a commonly used benchmarking methodology, used for example by Subramaniyan et al. (2018) to evaluate the accuracy of the ARIMA model. The naïve method simply takes the previous hour as the prediction for the next hour which in turn results in a prediction model which is possible to use for comparison.

2.4.2 Long-term bottleneck

When evaluating the accuracy of a bottleneck prediction algorithm it is not always the naïve method that is most successful. In some cases it can be beneficial to identify the long-term bottleneck and continuously predict the long-term bottleneck as the coming hours' bottleneck.

3
Methodology

In this chapter an overview is given over the work methodology, the reasons for the procedures and how the outcomes from the work were analysed. The methodology chapter aims to give the reader a structured view of how a machine learning model was developed and how different processing and analysing tools were applied. The overall methodology is illustrated in figure 3.1.

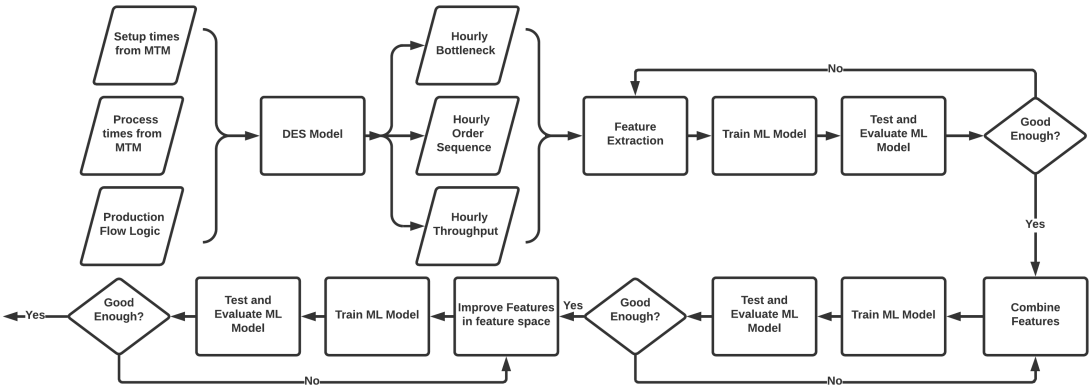

Figure 3.1: Overall methodology structure showing how the machine learning model was built.

Considering how the thesis was heavily reliant on data-mining, the Cross Industry Standard Process for Data Mining (CRISP-DM) (Wirth and Hipp, 2000) methodology was followed to a large extent for the part related to data mining in the project, visualised in figure 3.2 (Jensen, 2012). Both the methodology chapter and the result chapter will follow the structure of CRISP-DM.

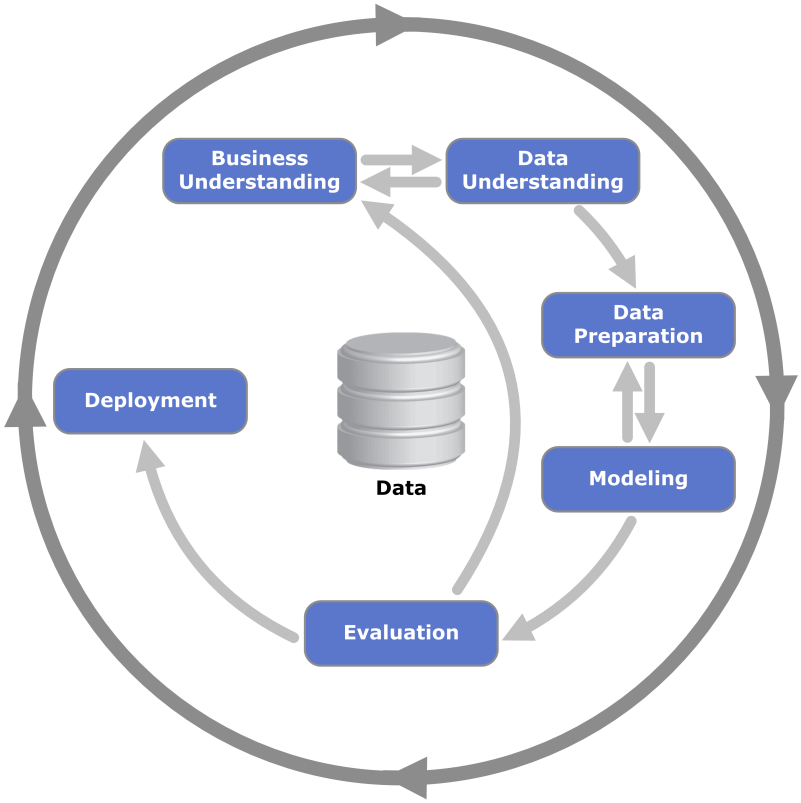

Figure 3.2: Illustration over the CRISP-DM methodology (taken from Jensen (2012))

3.1 Understanding the business

The first step of the project was to identify what was to be achieved and what was considered an acceptable result. It was important to assure that the project aimed towards a goal which would be value-adding in the industry. In order to aim the time and resources towards a desirable goal, an open dialogue was held with three different stakeholders:

1. Virtual Manufacturing, a consultancy firm within production development - niched toward, but not limited to, lean manufacturing, virtual tools and digitalisation.
2. Adient, a car seat manufacturer whose production line was used for the case study.
3. PDSVision, a provider of Product Lifecycle Management software and IoT platforms.

The purpose was to find a region of application where machine learning could add value through autonomous analyse of the production line, generating predictions which could inform the production leaders and engineers of potential issues in advance.

3.2 Data collection and understanding

Once the objectives for the project were set the data required to fulfill these objectives could be identified. The data required to train the machine learning model was hourly data regarding order sequence, throughput and bottlenecks. Out of these three parameters there were only one readily available, the order sequence. How many products that were produced every hour and where the bottleneck for the specific hour was located were unknown and had to be sampled.

3.2.1 Data sampling

Due to uncertainty concerning the amount of data required to train the machine learning model it was impractical to sample the data within the time frame of the project. Thus, the decision fell on a DES model which would provide a large degree of freedom in terms of data sampling and the size of the training set.

To be able to build the DES model, data describing the process times, setup times and down-times was needed. Additionally, a clear understanding about the production flow was required for the model to mimic the production line as closely as possible. However, the assembly stations were all manual labour which would be hard to measure accurately. Instead the decision fell on MTM-times which were utilised when the line was balanced, where both the process times and setup times were included. The rigidity and potential inaccuracy of the MTM-times compared to the real time data was noted but was omitted since there were no better alternatives to use.

Considering how the objectives, or goals, for the machine learning model had been decided upon previously, the DES model was programmed to sample data required to calculate the hourly bottlenecks and throughput. As for the bottlenecks there were many different alternatives for deriving the position of the bottleneck. The decision fell on the shifting bottleneck algorithm (Subramaniyan et al., 2016b) since it is responsive in short time intervals and is likely to be used in manufacturing in the future to continuously monitor the bottlenecks throughout the production run.

3.2.2 Delimitations in the data sampling

When building a DES model there are always a certain degree of assumptions and generalisations, which are important to keep in mind. However, some of the cases where a DES model deviates from the truth can be due to conscious decisions. In this project it was important to reduce the built-in randomness in throughput and bottlenecks in order to reduce the inherent variance. The following events were turned off prior to running and exporting the data for the machine learning modelling.

1. Breakdowns
2. Variation of process time due to speed variance of human labour
3. Rework of products not passing inspection
4. Setup time

The information regarding the breakdowns were that they occurred once every 12 hours and lasted for 17 minutes, generally for the whole production line, with an exponential distribution. This led to clear outliers roughly every 12th hour, which added no value at all to the machine learning model considering how the breakdowns could not be predicted.

Variation of the process times due to variance of human labour is very difficult to model accurately and considering how the production line was almost solely based on human labour, the error in modelling this part would be too great to give an accurate description of the production line.

Rework was based on a random distribution with a 3% rework rate. Certain extreme case hours could have far higher rework rate, while some had 0% rework rate. This skewed the result a great deal and would also have reduced the accuracy to a large extent.

Finally there was the setup time. This was not a random variable but rather depended on how many products had passed the station. However, an analysis was made where the DES model only built one variant for several hours at a time. With the setup time activated, a large variance in in throughput was noticed. Thus, setup was turned off in the simulation model.

3.3 Data preparation

It was decided that the order sequence produced for the hour would be used as raw input to the machine learning model, and instead let the machine learning model extract features, which can be seen as data preparation. However, in order to identify the bottleneck for each individual hour a fair amount of pre-processing had to be performed on the recorded states of the machines.

As mentioned in 2.1.1 there are several different choices when it comes to bottleneck detection and each production line has its own preferred and suitable model. In this case, the shifting bottleneck algorithm (Subramaniyan et al., 2016b) was applied mostly because it is responsive in short intervals, it can detect both primary and secondary bottleneck and it is based on data commonly found in a MES-database for a production line.

3.4 Machine learning modelling

When all the data was sampled from the DES model it was time to identify different methods of preparing the data to train the machine learning model. The output from the DES model, namely the hourly throughput, bottleneck and order sequence were the foundation for the features that the machine learning model would be trained with. With the order sequence as a base, new features could be extracted and combined.

The process of generating features was an exploratory and iterative phase in the project. It was difficult to foresee what was excessive, deficient or too elementary for the machine learning model to find a clear pattern. The process consisted of five main components - researching different types of features, implementing chosen feature type, training the machine learning model, testing the machine learning model and evaluating the machine learning model. The process is illustrated in figure 3.3, where the different sections of the process are included.

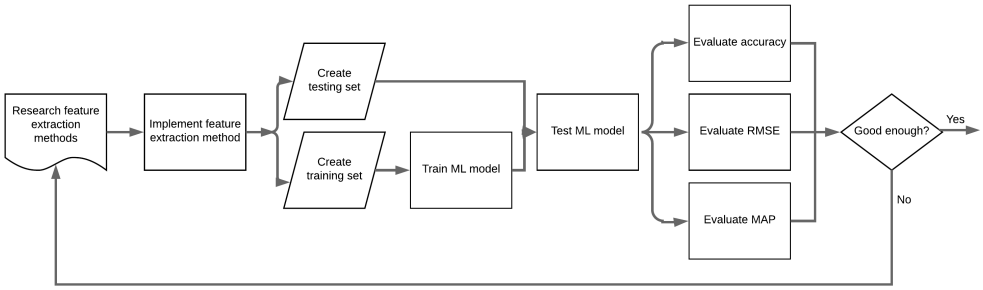

Figure 3.3: Illustration of sub-steps in methodology for building machine learning model.

3.4.1 Feature extraction methods

Feature extraction is the key part of applied machine learning. It varies heavily between different domains and there were no previous work done to simply follow. Thus, a lot of the different methods of feature extraction were generated through brainstorming or attempts to follow guidelines in other domains. In table 3.1 the different features implemented in this project are listed.

Table 3.1: Overview of the investigated features.

Name	Vector Length	Variable Type	Description
Order Sequence	100	Categorical	This was the raw output from the DES model, simply a list consisting of the variants produced in the specific order.
Histogram	12	Continuous	A histogram of the Order Sequence, where the amount of different variants is kept but the specific order is lost.
Machine Load	23	Continuous	The average process time for the hour for each individual machine or station.
Product Load	300-900	Continuous	Each individual variant in the order sequence was expanded into the individual process time at the stations with the largest variation in process time. The method was tested with different amounts of machines, hence the variation in length.

3.4.2 Evaluation of required training set size

The size of the data set needed to fully train a machine learning model varies heavily depending on its application and the size of the feature vectors. Since the length of the feature vectors varied, the evaluation was not conducted until the evaluation of the performance of the feature methods was final. Once the best feature method was identified, the best feature was used to evaluate the data size required. The test was performed by monitoring how the accuracy and statistical metrics changed for different data sizes. Starting with a very low data set size, where it was clear that it was not fully trained, and then slowly adding additional data to the data set resulted in a graph where the performance is stabilised after a certain amount of data.

3.4.3 Improving data in feature space

In machine learning it is common to use very large data sets in order to reach a desirable result. However, there are ways to make the patterns clearer in the feature space. To make the patterns more explicit, different methods were investigated to help the machine learning model as much as possible. Four different methods were implemented and tested to see if there were any possibilities to increase the performance. The methods are presented in table 3.2 where the aim and a short description for each method is given.

Table 3.2: Overview of implemented methods for investigating improvement of data in feature space.

Name	Aim	Description
Normalisation, Witten et al. (2017)	Make the patterns more profound.	Normalisation aims to bring the values to a range between 0 and 1 in order to reduce the magnitudes affect on the algorithm.
Exponential increment	Make the patterns more profound.	If the distance between two attributes was very small it could be beneficial to increase it with an exponential function.
Principle Component Analysis (PCA), Howley et al. (2006)	Reduce noise and feature dimensions.	PCA aims to reduce the feature vectors size while maintaining as much of the information as possible by sorting out attributes that has ow variation.
Augmentation techniques, LeCun et al. (1998)	Reduce bias toward over-represented classes and intervals	Augmentation attempts to generate additional, synthetic data for the underrepresented classes or intervals. There are several different methods of augmentation that has been applied - replicas, added noise, interpo lation and extrapolation

To properly assess whether the changes had any desirable effect on the accuracy, an iterative methodology was applied for it. By applying different processing methods in feature space to the developed features, the performance was compared prior to and after adding the processing step. The methodology is visualised in figure 3.4.

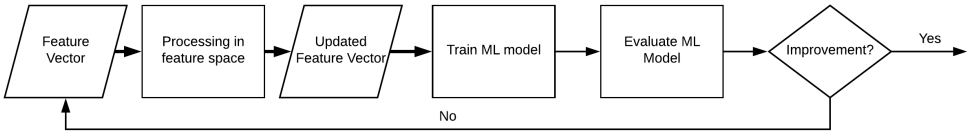

Figure 3.4: Illustration of methodology for applying and testing different processing methods in feature space.

3.4.4 Combining features

To attempt to increase the performance of the machine learning model different features were combined in order to see if pairing of two or more different features could enhance the model (Zhang et al., 2011). Considering the amount of different features available and processing methods in feature space implemented, this was a phase where a considerable number of feature vectors could be created. However, due to constraints concerning time and processing power some limitations had to be made.

One feature was combined with another feature to an extended input vector to the machine learning model. If there were no improvements to the results, the conclusion was that these two features were too similar to add additional value to each other. If any of the features could add value to two different features, all three of these were combined to evaluate if it could be improved even further. The process is illustrated in figure 3.5.

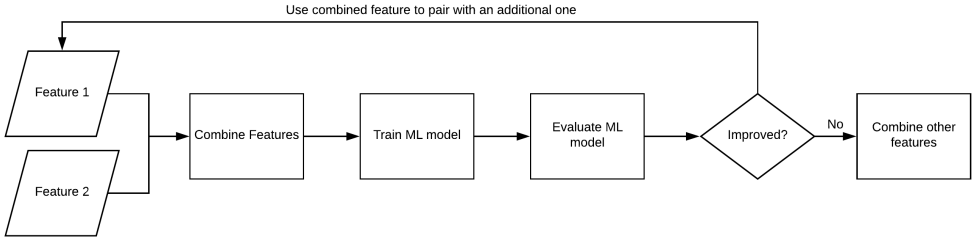

Figure 3.5: Illustration of methodology for combination of features when building machine learning model.

3.5 Evaluation and benchmarking of machine learning models

After the machine learning models were created through training with the different designed methods, they had to be compared and evaluated against each other. This was done by initially creating a scoring job on the models, where the testing data set was added and run through the models to acquire the predictions to be used for the evaluation.

The evaluation was performed with different methods depending on if the goal variable from the model was of the continuous or categorical kind. For the throughput prediction the naïve method was used for comparison while the bottleneck prediction used both the long-term bottleneck and the naïve method for comparison.

3.5.1 Continuous output evaluation

The evaluation of the models' continuous output was divided into three section. One section consisted of statistical plots which were used to get a visual overview of the results, and a second section which included statistical metrics and were used to get actual measures of the goodness of fit of the models, and finally a comparison was made of the performance of each model compared to the naive method, using the paired t-test to analyse the statistical significance.

Table 3.3: Overview of method and metrics used to evaluate and benchmark the continuous output, i.e. throughput, of the models

Statistical plots	Statistical metrics	Benchmark
Scatter plot	RMSE	Naïve method
Histogram plot	MAPE	t-test
-	Accuracy (Threshold)	-

3.5.1.1 Statistical plots

For the statistical plots, Excel was used to handle the data and generating the plots. The first step was to create a scatter plot over the predicted and the actual values. By adding a perfect fitted line for the actual values and a regression line for the trend of the data points, it was possible to get a sense of how well the models managed to find the pattern between the input and the output.

The second step was to create a frequency range for both the actual and the predicted values. With the frequency range, it was possible to construct two separate histograms which showed the distribution of the predicted respectively the actual values. This gave a better understanding of how well the models were able to copy the real distribution.

3.5.1.2 Statistical metrics

For the statistical metrics, the first step was to calculate and investigate the RMSE measure. The RMSE was a good measure to get a direct measurement of how the error was increasing or decreasing using different machine learning models with the same scale on the data. As a compliment to the RMSE measure, the MAPE measure was derived. This measure was more optimal to use when describing the result without knowing the scale of the data since it gave a percentage value of the error instead.

The models were also evaluated using three different thresholds for accuracy, i.e. having a maximum of 1, 0.75 and 0.5 product deviation between the actual value and the predicted value. This method of evaluation was used to get a percentage measure for the total accuracy and get a better overview of how the models' accuracy changed with more narrow thresholds.

3.5.1.3 Benchmarking with naïve method

As a final evaluation for the models the naïve method was utilised and the methodology for the benchmarking was taken from Subramaniyan et al. (2018). The naïve method assumed that the throughput of the previous hour would be the same as the current hour. By using the same thresholds for the naïve method as used for the accuracy calculation, it was possible to evaluate if the accuracy of the model could exceed the accuracy of using the naïve method. Comparing the mean accuracy between the naïve method and the investigated model does not work as a very suitable evaluation metric since it lacks information about the sensitivity of the two methods. Therefore, the paired t-test was utilised to derive a value called t_{val} as an evaluation metric for the performance between naïve and the tested model.

3.5.2 Categorical output evaluation

The evaluation of the models' categorical output was implemented by creating a confusion matrix (section 2.3.4) over the results from the models. From the confusion matrix it was possible to calculate and depict the total accuracy of each model, i.e. observe how well each model could predict the correct bottleneck station. Moreover, the precision was calculated to see how many of the predicted bottlenecks were the actual bottleneck station. Finally, the recall was derived giving an indication of the amount of the actual bottleneck station that was predicted accurately.

The naïve method was used for the categorical output as well, where the naïve method assumed that the bottleneck for the previous hour would be the same as the current hour. Then the accuracy from the confusion matrix was compared with the accuracy from the naïve method to see if the model was able to out-predict it.

3.5.3 Investigation of poor performance

If the machine learning models did not performed at a satisfactory level, meaning that neither the accuracy nor the statistical metrics were acceptable, a further analysis had to be made. The methods presented in this section all aimed to address a different hypothesis as to why a prediction model performed poorly.

3.5.3.1 Leveling categorical output distribution

If the distribution of the categorical output was very uneven, it was possible to test to level out the sample sizes of the output. By running a much longer simulation and creating a vast data set, data reduction principles were utilised to remove samples with the dominant category or categories. This created a more even distribution of data with more information related to the categories that were lacking beforehand.

3.5.3.2 Unsupervised learning to evaluate the pattern

To investigate whether or not there was a present pattern, k-nearest-neighbour was applied (Altman, 1992). This investigation could point the development in the right direction when attempting to improve the model, since this kind of learning can not be biased or have too large variance.

K-nearest-neighbour is an unsupervised machine learning method which aims to detect previously unknown patterns. By finding the nearest neighbours to the feature vectors representing a specific class, it was possible to investigate if the nearest neighbours belonged to the same class or not. In this case K was set to 10, meaning that the 10 nearest neighbours was found and identified as true or false.

3.5.3.3 Evaluation of suitability of bottleneck detection method

If the performance of the bottleneck prediction was poor there could be due to the complexity of the bottleneck detection method. In order to assess if this was the main issue with the prediction model a more simple, possibly more predictable bottleneck detection model was implemented. The bottleneck detection model should preferably use the same input data, which is why the active period percentage algorithm was chosen.

4
Results

The result chapter covers the information gained during the different steps in the CRISP-DM methodology. The evaluation step covers the result from different feature methods, improvements in feature space and combination of features. It aims to present the result corresponding to the time-line of the development of the machine learning model.

4.1 Business understanding

To fully understand what the industry was interested in, a deeper investigation to the matter was made. In this section the objectives and how they are evaluated are presented.

4.1.1 Defining the objectives

From the industry side, namely Virtual Manufacturing and Adient, of the project the main interest was on bottleneck identification rather than the throughput since it was crucial to understand the root cause of the problem rather than the consequences of said problem. Both of them in combination was of course the best case scenario. Here, throughput is defined as the amount of product that leave the production line during the specific hour.

Since the production strategy at Adient relies heavily on precise deliveries with a very short lead time, the order sequence is only known for a short time prior to production. This meant that the time frame of the machine learning models' prediction had to be even shorter. Thus, the time frame was set to one hour which also allows for an easy comparison to the desired takt-time.

From PDSVisions' point of view there was similar interest as for the industry side, since they were representing the potential customer of machine learning applications. However, PDSVision was also interested in comparing ThingWorx Analytics to a strong computational software. For this comparison MATLAB was chosen, which is a strong numerical computation software that has a multitude of tools for both IoT and machine learning.

It is important to define when the results are deemed successful. Since benchmarking with the naïve method is used to evaluate the machine learning models, the results from the models can be seen as successful if they surpass the accuracy and statistical metrics of the naïve method. Important to note is that the t-test value validates that the results does not happen by accident, but instead are due to a successful machine learning model.

4.2 Data understanding

Due to complications with extracting data from the MES-system at Adient the only available data was the MTM-times. The MTM-times alone was not sufficient to build a machine learning model. In order to fulfill the set objectives in the business understanding, both the bottleneck and throughput for each hour had to be generated through the DES model.

As for the input data to the machine learning model it was important to log the order sequence that was planned to be produced. However, some of the products started from the previous hour were still on the production line and had not yet left been finished. Thus, the previous hours' order sequence was also included in the input vector.

Since the shifting bottleneck algorithm was the most fitting bottleneck detection method for the specific production line, the state of each station had to be known for each second. This meant that a binary measure for each station was recorded every second, leading to 3600 measures for 20 different stations every hour. For the throughput objective, the products that left the production line were simply counted during the hour.

When analyzing the effects from different variants on the throughput it was evident that it varied a lot. By only producing one variant type at a time on the production line for several hours both the variation and the mean throughput for the specific variant type differed a great deal, see table 4.1. Further, it also gave information regarding the internal variation of the production line.

In order to be able to apply paired t-test, two assumptions about the data set was made. The first one was that the data was normally distributed. The second assumption was that the data was independent, which is a safe assumption seeing how all the data was generated through the same DES model.

Table 4.1: Overview of how each variant, V_i, vary in maximal, mean and minimal throughput per hour with setup times included. The probability for each variant to be produced is also included.

Variant	Max Output	Mean Output	Min Output	Probability (%)
V_1	74	72.85	72	17.5
V_2	74	72.85	72	4.18
V_3	90	88.83	87	0.39
V_4	90	88.83	88	2.39
V_5	89	87.24	85	7.01
V_6	90	87.24	86	3.27
V_7	89	87.24	85	0.48
V_8	89	87.28	85	1.30
V_9	92	90.04	88	19.26
V_{10}	93	90.04	88	10.35
V_{11}	92	90.04	89	28.08
V_{12}	97	95.95	94	5.81

What became evident in the evaluation of the internal variation was that the variation was too large. For example, variant 10 had a range from 88 to 93 products per hour. This variation heavily depended on the setup time and how they were aligned for each individual hour. Since the setup times were not dependant on the specific variants the setup times were removed, which reduced the internal variation heavily, see table 4.2

Table 4.2: Overview of how each variant, V_i, vary in maximal, mean and minimal throughput per hour with setup times excluded.

Variant	Max Output	Mean Output	Min Output
V_1	74	72.76	72
V_2	74	72.76	72
V_3	92	90.87	90
V_4	91	89.84	89
V_5	89	88.02	87
V_6	89	88.02	88
V_7	89	88.23	88
V_8	89	88.23	87
V_9	91	89.84	89
V_{10}	91	89.97	89
V_{11}	91	89.84	89
V_{12}	97	96.00	95

4.3 Data preparation

Out of the three data types sampled from the DES model there was only one that needed to be pre-processed at this stage - the stations' states. In order to detect the bottlenecks in the production line for each specific hour the stations' states was used as input. This algorithm generated both the primary and secondary bottlenecks, presented in the table 4.3.

Table 4.3: Example of the resulting distributions of the primary and secondary bottlenecks from the system.

Bottleneck Station	Primary Vol. (Dist.%)	Secondary Vol. (Dist.%) .
RSA-04	1672 (12.62)	9815 (74.04)
RSA-06	90 (0.679)	507 (3.824)
RSA-26	11181 (84.39)	1541 (11.62)
RSA-32	305 (2,302)	1193 (8.999)
RSA-13	0	200 (1.508)

As can be seen in table 4.3 the variation in primary bottlenecks was very low. With a variation that was so low, machine learning models were less likely to be successful, thus the focus of this thesis landed on secondary bottlenecks in the production line since there was a larger variation in that area. This could in turn result in a more interesting result.

As for the order sequence of the variants there was a lot of pre-processing. However, this type of pre-processing is often referred to as feature extraction in machine learning modelling, and will be found in individual sections for each feature method.

4.4 Modelling

During the modelling of the different feature methods there were some instances where data analysis had to be done to successfully model the features. Under this section the results of these analysis are presented.

4.4.1 Data investigation of order sequence

The order sequence method is the same as the order sequence logged for each individual hour in the DES model. However, since the previous hour plays an important role in the throughput a certain part of the previous hours order sequence had to be included as well. This extended order sequence was then used as the base for all other feature methods.

In order to know how much of the previous hours order sequence was to be included a pass-through-time analysis was performed. This resulted in an average of 26 minutes for a product to pass through the system. This would mean that almost half of the previous hour's orders were still in the system when the next hour started, and at least half the order sequence had to be included.

However, the products that had already left the production line would still have affected the state of the production line. If the product were slow there could be a lot of blockage in the system, resulting in a higher throughput for the coming hour. This lead to the decision to include the entire previous hour in the input to the machine learning model.

4.4.2 Data investigation of product load

In order to not include too many stations in the product load method, analysis had to be performed in order to see which stations had the largest variation. In table 4.4 the mean process time and the variance in process time for each station are displayed. It was evident that several stations had very large variance in process times for different variants, which most likely affected the throughput.

Table 4.4: Overview of the mean and the variance in the process time of each station (only stations with variance > 0 included).

Station	Mean	Variance
RSA-01	68.32	67.22
RSA-02	32.85	99.87
RSA-03	44.02	265.7
RSA-04	67.94	2.12
RSA-05	55.96	212.77
RSA-06	60.95	201.87
RSA-10	59.57	33.13
RSA-11	59.85	24.15
RSA-15	183.16	119.7
RSA-16	195	99.86
RSA-22	51.5	832.6
RSA-26	70.37	49.95
RSA-28	70.77	23.13
RSA-32	42.75	30.13

By choosing the six stations with the largest variance in process time the feature length expanded from 100 to 600. The added size increased the training time required, but by assuring that all the information was value-adding, the result could be improved. To evaluate how many stations' process time should be included, it was increased from one station up to nine stations, with the best performance achieved with six stations included.

4.5 Evaluation

The evaluation section of the result chapter covers the machine learning evaluation. In order to get a complete overview of the process, it covers the result from each of the different feature methods, an analysis of the amount of data required, the improvements in feature space, the combination of feature methods and an overview of the bottleneck prediction.

The feature methods developed throughout the project were all compared with the naïve method which was used as the present state of predictions. Moreover, some of the subsections in this chapter does not only present the result of the prediction model, but also an analysis done during the implementation stage. After all of the results for the different feature methods have been presented a summary is displayed to compare the different methods.

In figures 4.1-4.5 scatter plots can be observed over the results from the different feature models. The predictions that are assigned a circle signifies being inside of the threshold 1, and the predictions assigned a cross are outside of the threshold. The dotted line represents the so called "perfect line", i.e. if all predictions followed this line it would be a perfect result. The dash-dotted line signifies the regression line and describes how well the trend of the predictions fit the "perfect line".

4.5.1 Throughput prediction

The throughput prediction was overall successful. Although there was a bias toward the more common values, the machine learning model managed to accurately predict a large amount of the samples and outperformed the naïve method by far in every way. Further, the t-test score for every method shows that there is a clear statistical difference between the different methods and the naïve method.

In this section the result for each individual feature method is presented, with a summary in the end and an evaluation of the data size required. Further, results for improvement in feature space and combination of features are also presented.

4.5.1.1 Naïve method

The naïve method was used as a present state analysis that every other feature method was compared to. Its performance on this specific production line was very poor. As can be seen in figure 4.1 the scatter plots shows a neglectable linear correlation between the observation and prediction.

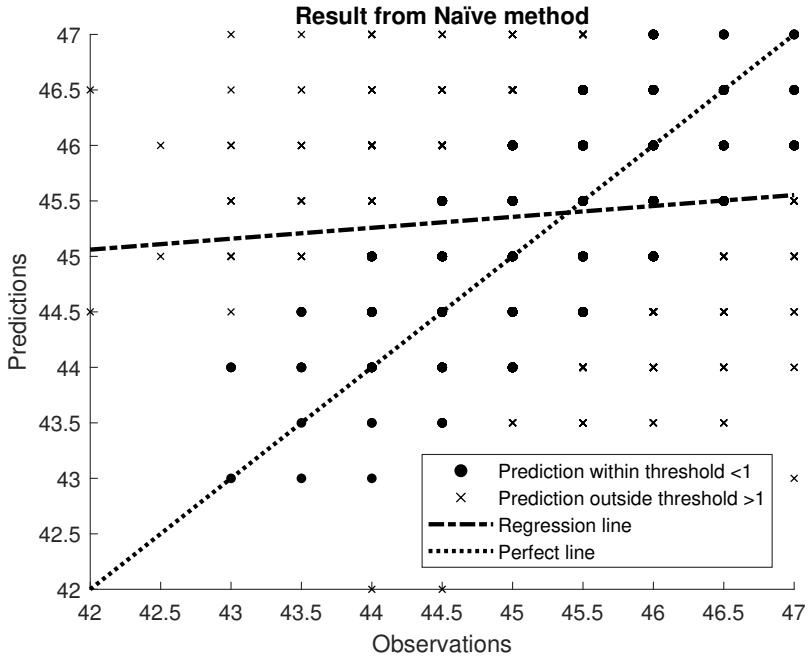

Figure 4.1: Scatter plot over observations vs. predictions for the result from the Naïve method.

Further, the accuracy and the statistical metrics presented in table 4.5 are the baseline used in this thesis for comparison when evaluating the developed feature methods. However, the accuracy of 76.09% could be regarded as fairly high with the low correlation of the trend line in mind. This is due to the distribution of the observations, which are over represented towards the middle of the range.

Table 4.5: Overview of the results from the statistical metrics for the Naïve method.

Statistical metrics	Naïve Result
Accuracy 1 (%)	76.09
Accuracy 0.75 (%)	48.58
Accuracy 0.5 (%)	48.58
RMSE	1.118
MAPE (%)	1.894

4.5.1.2 Histogram

The histogram feature method was based on how many of each variants were scheduled to be produced. As can be seen in figure 4.2, the correlation between the prediction and the observation for the histogram feature was far better than the correlation of the naïve method in figure 4.1. The slope was now steeper, but the range of the predictions for each individual observation value was still far too large to be considered acceptable.

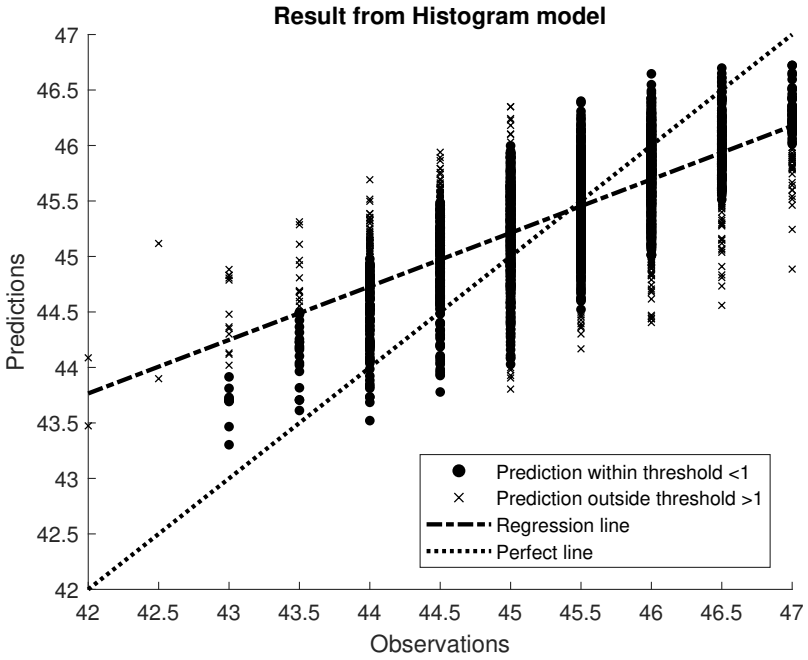

Figure 4.2: Scatter plot over observations vs. predictions for the result from the Histogram model.

The accuracy, which is presented together with the statistical metrics in table 4.6, was increased to 89.97% which was considerably better than the accuracy of the Naïve method. It was also evident that both MAPE and RMSE had been reduced a considerable amount. Further, the value from the t-test reveals that there was a significant statistical difference as compared to the naïve method, showing that the model found a pattern and it is not caused by chance.

Table 4.6: Overview of the results from the statistical metrics for the Histogram model compared to the Naïve method.

Statistical metrics	Histogram Result	Naïve Result
Accuracy 1 (%)	89.97	76.09
Accuracy 0.75 (%)	78.53	48.58
Accuracy 0.5 (%)	59.72	48.58
RMSE	0.599	1.118
MAPE (%)	1.048	1.894
t_{val} (Naïve)	22.04	-

4.5.1.3 Order sequence

The order sequence method was the input data with no pre-processing applied. A method which, as opposed to the histogram method, kept the order of the variants. In figure 4.3 it is evident that the machine learning model is not only performing better than the naïve method but also better than the histogram method. The trend line gets closer to the perfect line while also keeping the range of the predictions for each observation closer than the range of the histogram method.

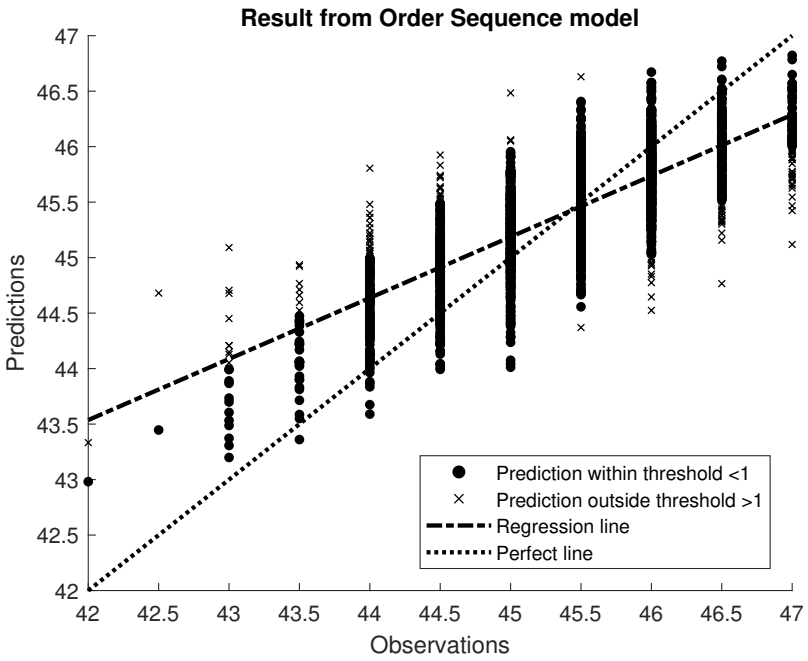

Figure 4.3: Scatter plot over observations vs. predictions for the result from the Order Sequence model.

As can be seen in table 4.7, the accuracy of the order sequence method was improved to 94.59%. This was an improvement of almost 20% more than the naïve method, but also almost 5% more than the histogram method, which is a significant improvement which, showed that the specific order of the variants was important. Further, the MAPE and RMSE were reduced even further which complements the observation concerning the reduction in range for the predictions for each specific observation. The t-test value was even greater, indicating that the model deviated from the naïve model even more.

Table 4.7: Overview of the results from the statistical metrics for the Order Sequence model compared to the Naïve method.

Statistical metrics	Order Sequence Result	Naïve Result
Accuracy 1 (%)	94.59	76.09
Accuracy 0.75 (%)	86.08	48.58
Accuracy 0.5 (%)	67.09	48.58
RMSE	0.520	1.118
MAPE (%)	0.907	1.894
t_{val} (Naïve)	26.38	-

4.5.1.4 Machine load

As can be seen in figure 4.4 the machine load method was still better than the naïve method and fairly equal to the histogram method, but a step back as compared to the order sequence method. The trend line shows a correlation, but since the specific order of the variants is once again lost this method had a large resemblance to the histogram method.

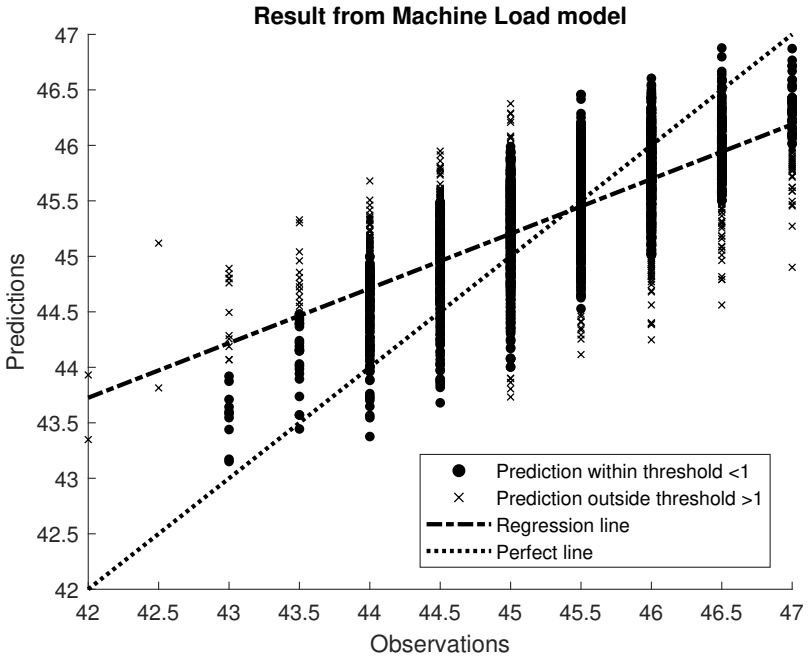

Figure 4.4: Scatter plot over observations vs. predictions for the result from the Machine Load model.

The accuracy, displayed in table 4.8, shows that the method was very similar to the histogram methods' accuracy. Further, both MAPE, RMSE and the t-value were basically identical to the statistical measures of the histogram method. This was interesting since it meant that the machine learning model understood the correlation between the histogram method and the sum of the process times for each machine. The method is still remaining far better than the naïve method but beaten by roughly 5% by the order sequence method.

Table 4.8: Overview of the results from the statistical metrics for the Machine Load model compared to the Naïve method.

Statistical metrics	Machine Load Result	Naïve Result
Accuracy 1 (%)	90.25	76.09
Accuracy 0.75 (%)	79.31	48.58
Accuracy 0.5 (%)	59.95	48.58
RMSE	0.599	1.118
MAPE (%)	1.048	1.894
t_{val} (Naïve)	22.02	-

4.5.1.5 Product load

As can be seen in figure 4.5, product load has a regression line which is fairly close to a perfect line, meaning that the linear correlation between the prediction and observation is high. Compared to the naïve method product load performed significantly better, but also compared to the histogram method, machine load method and order sequence method. The slope of the trend line is far closer to the perfect line compared to any of the other methods and the range for the predictions for each specific observed value is also reduced.

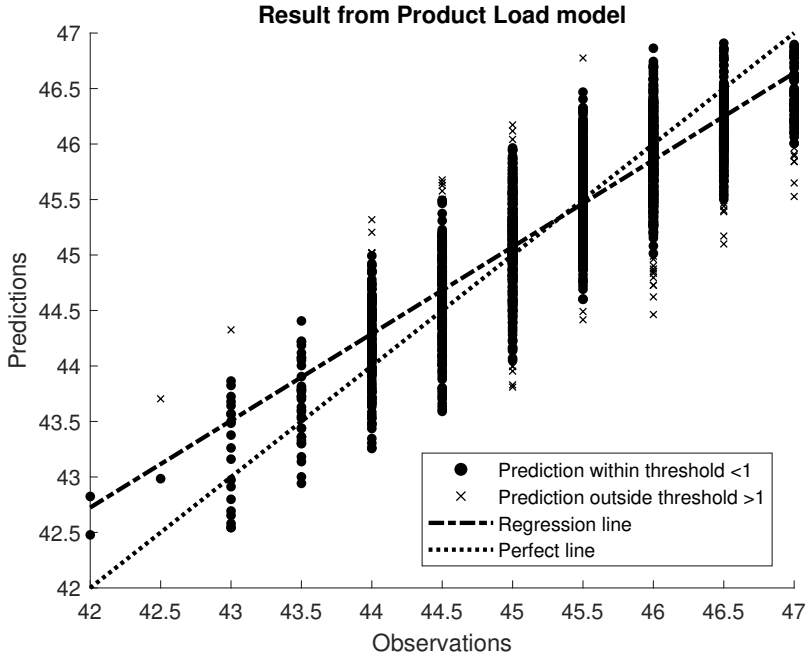

Figure 4.5: Scatter plot over observations vs. predictions for the result from the Product Load model.

The accuracy and statistical metrics for the product load method can be seen in table 4.9. It is evident that the accuracy of 97.80% is far superior to the accuracy of 76.08% for the naïve method, but also the accuracy of the other feature methods. Although order sequence has an accuracy 1 that is fairly close to the product load method, the accuracy 0.75 of 92.73% for the product load method and 86.08% for the order sequence method shows that there is still a large difference in performance. The t-test value is also very different from all other methods, indicating that the model deviates largely from the other models. In addition to have a superior accuracy the error measurements are far better than of any of the other methods.

Table 4.9: Overview of the results from the statistical metrics for the Product Load model compared to the Naïve method.

Statistical metrics	Product Load Result	Naïve Result
Accuracy 1 (%)	97.80	76.09
Accuracy 0.75 (%)	92.73	48.58
Accuracy 0.5 (%)	75.28	48.58
RMSE	0.429	1.118
MAPE (%)	0.755	1.894
t_{val} (Naïve)	31.37	-

4.5.1.6 Overview of throughput prediction

In table 4.10 the statistical measurements for the naïve method and all the feature methods are presented. It is once again clear that the product load method is superior, while order sequence is fairly close. The t-test value reveals that the statistical difference is significant for all of the methods since it lies far outside of the confidence interval. It also showcases that the machine load method and histogram method are identical in all aspects.

With all of the information presented for the evaluation of feature methods it was clear that product load was the best feature method to train the machine learning model. Thus, for the next step of the process the product load method was used while the other methods were not evaluated with the feature space improvements.

Table 4.10: Summary of the overall results from running each feature method in ThingWorx.

Statistical metrics	Histogram	Order Sequence	Machine Load	Product Load	Naive
Accuracy 1 (%)	89.97	94.59	90.25	97.80	76.09
Accuracy 0.75 (%)	78.53	86.08	79.31	92.73	48.58
Accuracy 0.5 (%)	59.72	67.09	59.65	75.28	48.58
RMSE	0.599	0.520	0.599	0.429	1.118
MAPE (%)	1.048	0.907	1.048	0.755	1.894
t_{val} (Naïve)	22.04	26.38	22.02	31.37	-

4.5.1.7 MATLAB comparison

MATLAB was used for each feature method to make a comparison between using ThingWorx (ensemble technique) and MATLAB (single algorithm). The Regression Learner toolbox was used and by running a test with all the included algorithms in the toolbox, the linear regression algorithm was deemed the one with best accuracy.

In table 4.11, the results from running each feature method in MATLAB with linear regression can be observed. The accuracy and statistical metric are very similar between the two software. ThingWorx has a slightly higher accuracy on the lower thresholds as well as a bit lower RMSE and MAPE for the feature method of product load. Important to note is that MATLAB has better performance regarding the order sequence method, which is the only method using categorical data as input. This could indicate that MATLAB is superior to handling categorical data, although a further investigation on the matter should be made to get conclusive evidence of this.

Table 4.11: Overview of the MATLAB results for each feature method using Linear Regression algorithm.

Statistical metrics	Histogram	Order Sequence	Machine Load	Product Load
Accuracy 1 (%)	90.25	97.16	90.20	97.75
Accuracy 0.75 (%)	78.67	90.11	78.30	91.16
Accuracy 0.5 (%)	59.40	73.13	59.67	74.46
RMSE	0.603	0.458	0.603	0.443
MAPE (%)	1.058	0.800	1.058	0.771
t_{val} (Naïve)	21.76	29.85	21.77	30.76

4.5.1.8 Evaluation of data set size for throughput prediction

The result from evaluating how the size of the data set effects the result from the model can be seen in figure 4.6. The observation made was that the accuracy of the model reaches a steady state at around 7000 hours of data which corresponds to 1.29 years of data, assuming production 24 hours a day for 226 days a year. However, the accuracy was at an acceptable level at lower amounts of data as well.

Although accuracy is an interesting measurement when observing the data size for building the machine learning model, the goodness of fit metrics also needs to be evaluated. In figure 4.7 it is evident that the RMSE and MAPE reaches a steady state after 11000 hours of data, corresponding to two years of data as compared to the 7000 needed to reach steady state for the accuracy. This means that even though the accuracy does not improve, the error margin keeps improving up until 11000 hours.

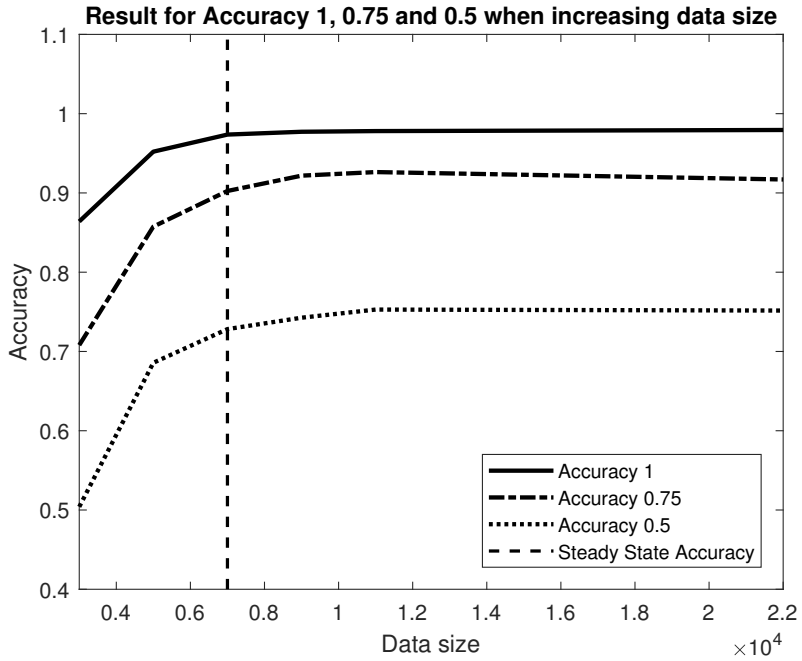

Figure 4.6: Figure showing how the accuracy is affected by increasing the data set size.

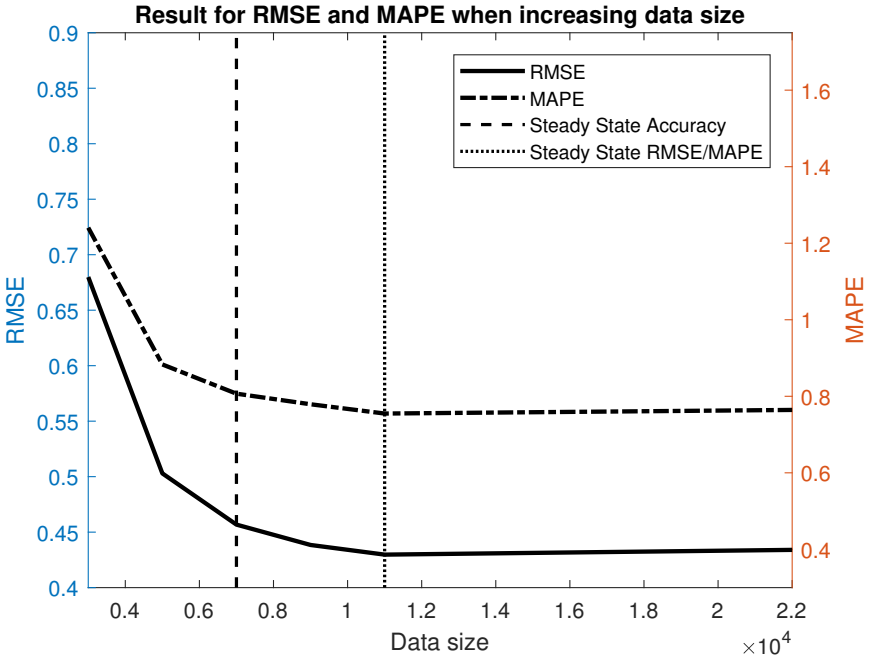

Figure 4.7: Figure showing how RMSE and MAPE is affected by increasing the data set size.

4.5.1.9 Improvements in feature space

As can be seen in table 4.12 the attempts to improve the data in feature space were unsuccessful. The accuracy as well as the statistical metrics were at less satisfactory levels. The only case where a slight improvement was noticed was for accuracy 1 when using augmentation. However, the difference was so small it was considered insignificant.

Table 4.12: Overview of the results from running each improvement in feature space in ThingWorx.

Statistical metrics	Product Load	Normalization	PCA	Augmentation
Accuracy 1 (%)	97.80	97.57	96.70	97.89
Accuracy 0.75 (%)	92.73	91.44	88.92	90.93
Accuracy 0.5 (%)	75.28	74.78	71.94	74.14
RMSE	0.429	0.439	0.469	0.443
MAPE (%)	0.755	0.769	0.824	0.781
t_{val} (Naïve)	31.37	30.89	29.12	30.57

4.5.1.10 Combination of feature methods for throughput prediction

In some cases, especially when using ensemble techniques, it was interesting to try different types of inputs. Since different algorithms prefer different types of data it could enhance the performance. In this project the investigated combinations were not successful. The only case where a combination of two different feature methods was beneficial was when the order sequence method was combined with the histogram method. The order sequence method had 94.59% on its own and the histogram method had 89.97% on its own, while the combination resulted in 96.61% as can be seen in table 4.13. However, this did not exceed the performance of the product load method on its own but shows the value that combinations has the possibility add.

Table 4.13: Overview of the results from running each combination of feature methods in ThingWorx (OS: Order Sequence, H: Histogram, PL: Product Load).

Statistical metrics	Product Load	OS + H	OS + H + PL	PL + H
Accuracy 1 (%)	97.80	96.61	96.52	97.71
Accuracy 0.75 (%)	92.73	89.56	88.74	92.03
Accuracy 0.5 (%)	75.28	70.52	72.58	75.46
RMSE	0.429	0.468	0.468	0.435
MAPE (%)	0.755	0.826	0.818	0.765
t_{val} (Naïve)	31.37	29.11	29.28	31.04

4.5.2 Overview of the bottleneck prediction

The bottleneck prediction was overall not very successful. There was a strong bias toward a certain station and the prediction would rarely beat the performance of simply predicting the same long-term bottleneck station all the time. In this section an overview of the results for the feature methods is presented.

The naïve method for the bottleneck prediction did in this case have 58.67% accuracy. As can be seen in the confusion matrix (figure 4.8) the result is poor. Important to note here is that if the prediction is made to always be the long-term bottleneck, i.e. station RSA-04, then the accuracy would be 74.55%, which in turn would be a better comparison to use than the naïve method in this case.

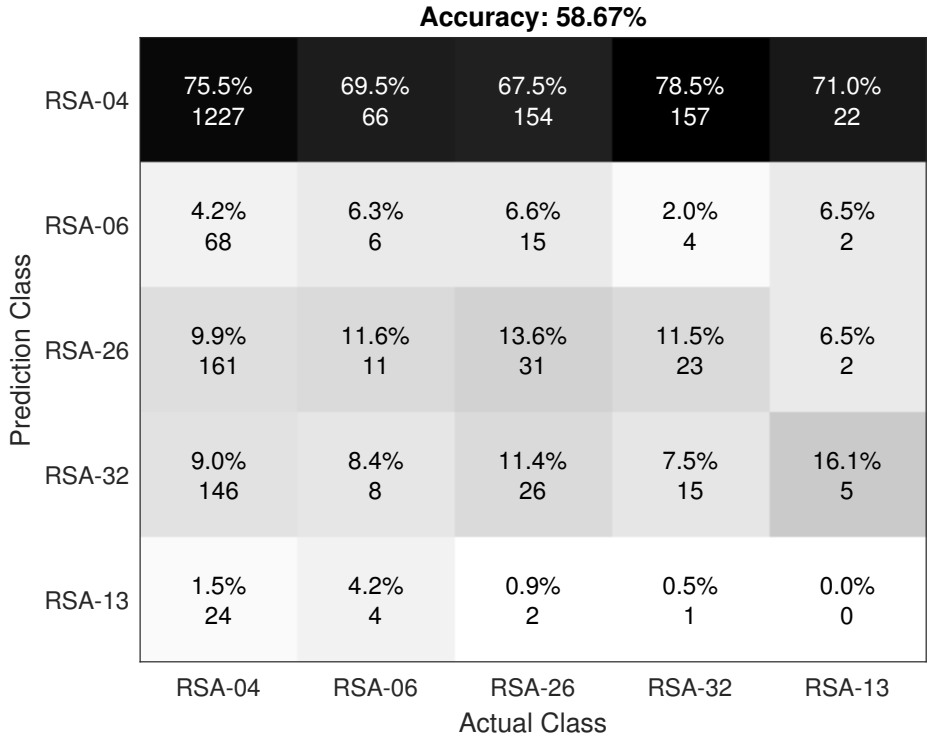

Figure 4.8: Confusion matrix with classification results for the Naïve method indicating the total accuracy, the distribution of classifications and the prediction percentage of the total of each actual bottleneck station.

In figure 4.9 the resulting confusion matrix for the product load method is presented. The confusion matrices for histogram, order sequence and machine load looks very similar to each other, where every method seem to only be able to predict the dominant long-term bottleneck well but do not manage to classify the other bottlenecks at all.

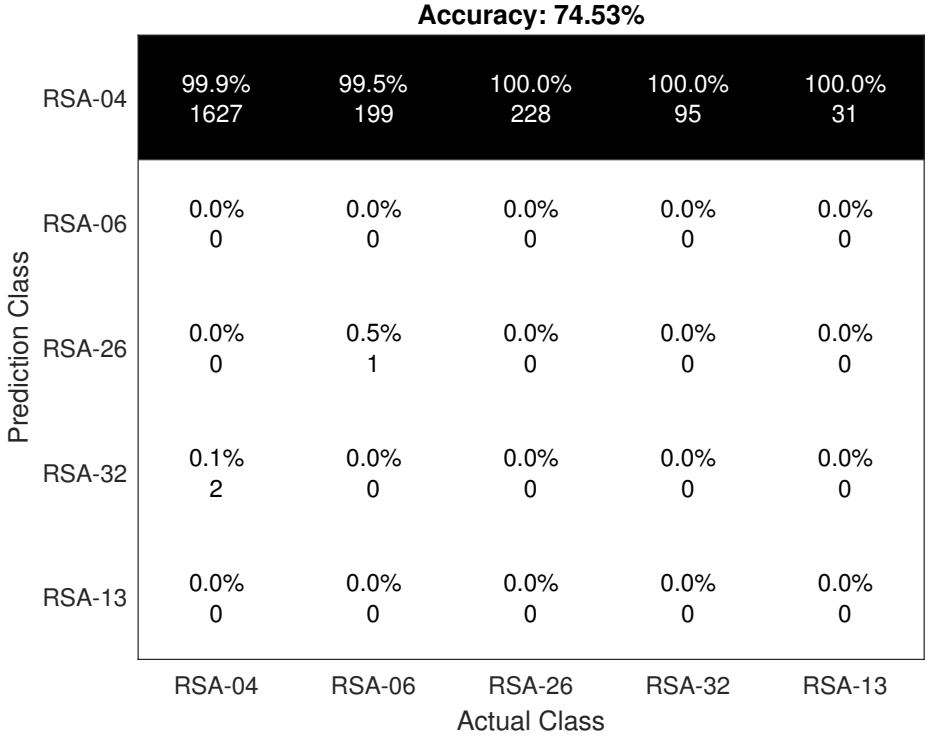

Figure 4.9: Confusion matrix with classification results for the Product Load method indicating the total accuracy, the distribution of classifications and the prediction percentage of the total of each actual bottleneck station.

In table 4.14 an overview of the accuracy, precision and recall is presented for each of the feature methods. As can be observed from the results in the table, the accuracy were very similar between the different feature methods, with none being able to exceed the long-term bottleneck prediction accuracy. The resulting recall demonstrates that the models from the feature methods had very high recall for bottleneck station RSA-04 and zero or very low recall for every other bottleneck. Furthermore, the resulting precision were very close to the total accuracy for the different methods, confirming that the total accuracy was solely dependent on predicting RSA-04 as the bottleneck.

Table 4.14: Summary of the resulting evaluation metrics from the confusion matrices with the total accuracy, precision and recall for every predicted bottleneck station when running each method in ThingWorx.

Metrics %	Histogram	Order Sequence	Machine Load	Product Load	Naive	L-term
Precision:						
RSA-04	74.60	74.93	74.58	74.63	75.46	74.55
RSA-26	0	45.93	0	0	13.60	0
RSA-06	0	0	0	0	6.320	0
RSA-32	0	13.33	0	0	7.500	0
RSA-13	0	0	0	0	0	0
Recall:						
RSA-04	99.69	98.43	99.75	99.88	75.46	100
RSA-26	0	4.820	0	0	13.60	0
RSA-06	0	0	0	0	6.320	0
RSA-32	0	1.000	0	0	7.500	0
RSA-13	0	0	0	0	0	0
Accuracy:	74.39	73.98	74.44	74.53	58.67	74.55

4.5.3 Evaluation of the level of distribution

Since the level of distribution of the output categories used to train the model was very uneven, a product load model was trained with an evenly leveled distribution of bottleneck categories. The resulting confusion matrix can be observed in figure 4.10. The result reveals that there still is a bias towards the same station as before, RSA-04, and therefore the accuracy was similar to the previous model results.

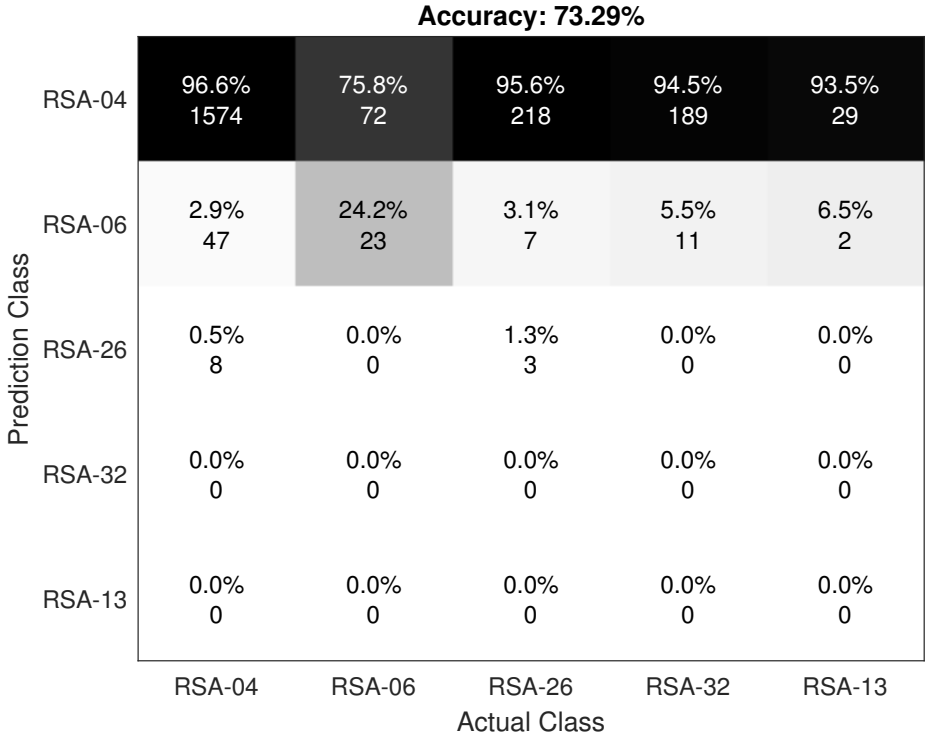

Figure 4.10: Confusion matrix with classification results from Product Load with a leveled distribution of bottlenecks, indicating the total accuracy, the distribution of the classifications and and the prediction percentage of the total of each actual bottleneck station.

4.5.4 Evaluation of pattern in bottleneck prediction

In order to evaluate if the model was too biased toward a specific bottleneck or if the pattern was too weak, an unsupervised clustering was performed. When finding the 10 most similar feature vectors, called neighbours, to the feature vector of a falsely classified bottleneck, there were never more than five neighbours belonging to the correct bottleneck. Further, it is evident in figure 4.11 that the most common amount of correctly identified neighbours was 0 or 1 out of 10 neighbours. This was a very poor performance and indicate that the pattern between the features and the output of the machine learning model was basically non-existent.

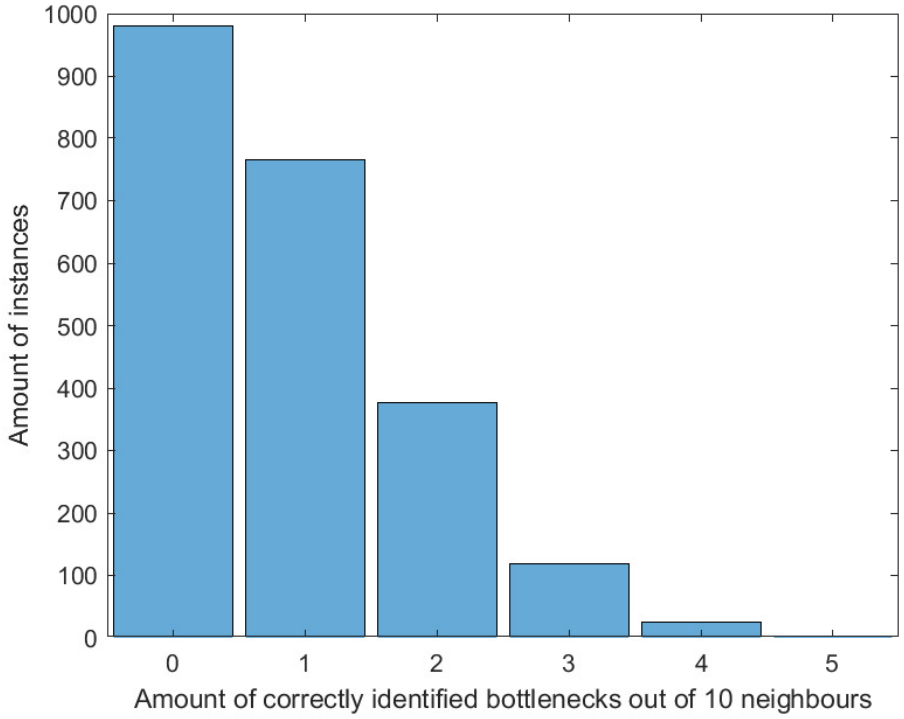

Figure 4.11: Figure displaying the histogram of true prediction using unsupervised learning.

4.5.5 Evaluation of suitability of bottleneck detection method

To investigate whether or not the shifting bottleneck detection methodology was the main reason for the poor performance, and missing pattern, a simpler bottleneck detection method was implemented, the active period percentage. The detected bottlenecks changed on a few station compared to the detection from the shifting bottleneck detection method, see section 4.5.2. However, it is clear for figure 4.12 that the prediction did not improve.

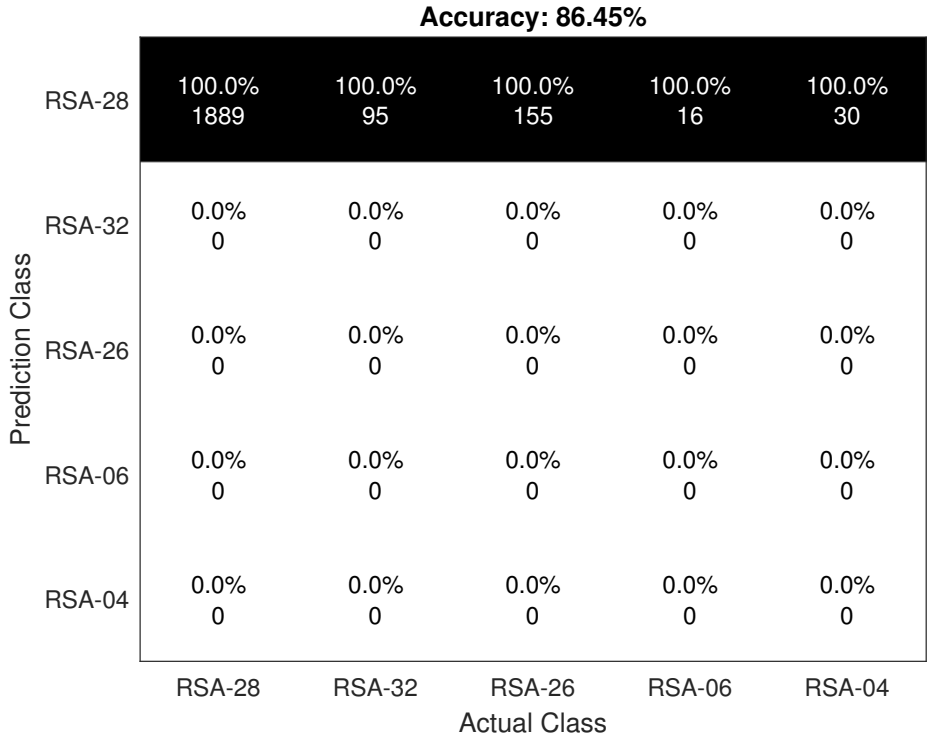

Figure 4.12: Confusion matrix with classification results from Product Load and Active Period Percentage method indicating the total accuracy, the distribution of the classifications and the prediction percentage of the total of each actual bottleneck station.

5
Discussion

The discussion chapter covers both the successful and unsuccessful results of the project, attempting to explain the causes and what can be gained from the results. The chapter is divided in three sections, throughput prediction, bottleneck prediction and benefits gained from the project.

5.1 Discussion on throughput prediction

The project can be considered successful for the throughput prediction. The machine learning model persistently outperforms the naïve method for all of the different statistical metrics and accuracy thresholds. This proves that machine learning can fruitfully be applied to a larger production line on a system level, and can accurately predict how the variant mix affects the productivity. The machine learning model might not be able to predict the exact throughput but it is accurate enough to give a strong indication on whether the order sequence is difficult or manageable to produce.

It is clear that there is a strong connection between the variant sequence and the throughput, and that the pattern can be recognised by the machine learning model. Since both the histogram method and the machine load method performed poorly compared to the product load method and order sequence method, the assumption can be verified that the distinct order of the variants is very important to accurately predict the throughput for an individual hour.

Since a DES model hardly can replicate the reality with complete accuracy, this type of application would be preferred to be based solely on historical data, something Subramaniyan et al. (2018) makes very clear. However, the analysis of data size required to properly train a machine learning model suggests that the machine learning model is not fully trained prior to one year. Assuming that changes are made continuously in a production environment the historical data would not describe the present. This means that a DES model would be needed to be continuously updated to generate data to train the machine learning model after each major change, which is a drawback in terms of accuracy but still a necessity due to the size of the historical data. One thing that Subramaniyan et al. (2018) had to do throughout the development was cleaning the data and preprocessing it. In a more controlled environment, such as DES, the time consuming step of data cleaning of the generated data can be avoided. For SMEs it can be very valuable to save both

the money and time of these tasks if the companies lack the resources to continuously analyse the data.

It is important to understand that a DES model is, as aforementioned, a controlled environment where certain disturbances are removed, and the behaviour can be more deterministic than stochastic in some cases. By creating the DES model to begin with, many of the behaviours and patterns that possibly could be detected by a machine learning model. could also be detected by the person building the DES model. However, there are patterns and dependencies within a production line that are too complicated to analyse manually even if a DES model is created. This is where the combination of a DES model and machine learning is a viable tool. If the DES model is believed to be accurate enough to base the balancing of a production line on, it should also be accurate enough to base a continuous evaluation of said balancing on it.

One thing that has not been tested in this thesis is how data augmentation would work on a training set consisting of less data than what is required to fully train a model. The possibility of using synthetic data to improve an under-trained model is something that is suggested as future research.

Compared to previous, similar studies (table 2.3) it is clear that the result for this machine learning model is within a reasonable interval in terms of the statistical evaluation metrics. It is difficult to compare models using different machine learning techniques, features and production systems but MAPE makes for an interesting comparison. The MAPE gives an indication on how closely the results lie. The model from this project with an MAPE of 0.755% is lower than the GRNN model (Samattapapong and Afzulpurkar, 2016) which had 1.1%, while it is higher than the ANN & ARMA (Pandian and Ali, 2013) which had 0.30% and 0.34% respectively. This indicates that the model is well built and the result lies within the range of the two previous studies. Moreover, previous research within the field have not presented the amount of data required to get acceptable accuracy, meaning that this project has pushed the research further forward in this aspect as well.

As for the development phase of this type of machine learning model there is a clear advantage of using ThingWorx ensemble technique compared to the single algorithms used in MATLAB. Depending on the data type of both the features and output of the machine learning model, different algorithms are preferred. In MATLAB there is a need to continuously evaluate that the best algorithm is used, while ThingWorx is evaluating it internally while building the model. This is something that the authors feel was very beneficial throughout the project.

5.2 Discussion on bottleneck prediction

Unlike the throughput prediction, the bottleneck prediction did not have a very good accuracy. Although all feature methods beat the naïve method, it is questionable if the information gained from this machine learning model can actually be value-adding at this stage.

The first assumption made to explain the poor classification rate of the machine learning models was that the bias towards one specific category of bottleneck was caused by very uneven distributions of categories in the data set. However, after testing leveling out the distribution for a training set and creating a new trained model, the result indicated that the bias did not come from an uneven distribution. Therefore, an assumption could be made that there might be a weak pattern in the classification method causing the poor performance.

In order to evaluate the assumption of a weak pattern in the classification method, the KNN method was applied. The result from the KNN evaluation confirms the hypothesis since the algorithm failed to find a pattern between the feature vector and the bottlenecks. This method of evaluation of supervised learning is, to the authors best knowledge, previously unused. However, both supervised learning and unsupervised learning aim to find a pattern in the input data, but the unsupervised learning can not be biased which means it can be used to evaluate both of the previous the hypotheses. The authors believe that using independent machine learning algorithms to evaluate the same pattern is an effective method to evaluate if there is a pattern within a data set.

The lack of a pattern could be due to the complexity of the shifting bottleneck algorithm. While it is a very accurate bottleneck detection method, small differences can lead to a large difference in the result. Idle times as small as one second could be the difference between a machine being denoted as the bottleneck or not. Thus, the next test was to change the bottleneck detection method to a simpler method, average active percentage. However, the result did not improve which means that the bottleneck detection method was not the main issue with the prediction model, while it is still possible that it complicates the matter.

The previous analysis leads to the assumption that it is the complexity of the classification methods that is the issue. There could be a mismatch between the input data and the classification method leading to the poor performance. Thus, it could be advantageous in future research to attempt to convert the output variable to continuous instead of categorical to avoid the issue. Instead of trying to predict the bottleneck, the bottleneck indexes for each machine of interest could be predicted and compared, which brings the issue back to previous research such as the ARIMA (Subramaniyan et al., 2018) and ANFIS (Cao et al., 2012) models that both predict an index rather than a category.

5.3 Benefits from the project

There are several key aspects where the company can benefit from this research. The first gain is the unsurprising knowledge that completely releasing the control of the variant sequence can pose considerable drawbacks. This project demonstrates that there is a potential for a large upside if the order sequence is controlled. Considering how all of the products are delivered to the same place, and are delivered in batches rather than one by one, it can be beneficial to rearrange the sequence for the products that are shipped together.

The productivity in a production line depends on several foreseen and unforeseen events, such as setup times, down times, uneven balancing, uneven speed of manual labour, etc. That the productivity throughout a production run will vary is very probable. However, to attempt to predict how all of the parameters will differ is a complicated task when they are grouped together. By separating them and focusing on a single parameter the problem can be narrowed down. This is what has been done in this project, a focus on how the balancing will affect the throughput. Rather than applying this machine learning model to predict the actual throughput it could be converted to predict a productivity index for how well the production line is balanced regarding the specific variant sequence.

Future research on the matter could be to attempt to balance and optimise the variant sequence for a set time interval. This could be done by attempting to implement a machine learning model that has the variant distribution as input and an optimised variant sequence as output. However, it is unclear if this task is currently too complicated for today's machine learning algorithms.

6
Conclusion

It is clear that a pattern between the order sequence and the throughput can be recognized on a system level in manufacturing. With an accuracy of 97.8%, a MAPE of 0.755% and RMSE of 0.429 the machine learning model built in this project is deemed successful. This shows that machine learning is mature to take a step up from the single-machine outlook that is currently the focus in manufacturing, to a more holistic view. The throughput prediction strongly links how the variant sequence will affect the productivity for the coming hour, clearly displaying that it can determine when there will be losses due to balancing issues or not. However, to optimally train a predictive model two years of data is required, which leads to the requisite of a DES model in cases where continuous changes are made to the production line.

Although the bottleneck prediction did not achieve a desirable result, it is widely acknowledged that there is a strong relation between bottlenecks and the throughput in a production line. This leads to the assumption that since there is a pattern for the throughput prediction it should also be possible to find a pattern concerning the bottlenecks in the future. The poor performance is due to unsuitable algorithms and lack of conclusive data for the bottleneck prediction and will require further research to actually be beneficial to the industry.

Finally, it has been proven that the IoT platform ThingWorx not only has the capabilities comparable to a strong numerical computing software such as MATLAB, but also offers a fair amount of support in deciding the optimal machine learning algorithm to use. This will be crucial in the era of digitalization and the next manufacturing paradigm, Industry 4.0, where IoT and Big Data are growing domains.

Bibliography

Agresti, A. (2013;2014). *Categorical data analysis(3rd;3; ed.)*. Wiley, Hoboken, NJ.

Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185.

Armstrong, J. S. and Collopy, F. (1992). Error measures for generalizing about forecasting methods: Empirical comparisons. *International Journal of Forecasting*, 8(1):69–80. doi:10.1016/0169-2070(92)90008-W.

Azizi, A. (2016). Integration of seasonal autoregressive integrated moving average and bayesian methods to predict production throughput under random variables. *Journal of Mechanical Engineering and Sciences*, 7:1236–1250. doi:10.15282/jmes.7.2014.23.0121.

Barnston, A. G. (1992). Correspondence among the correlation, rmse, and heidke forecast verification measures; refinement of the heidke score. *Weather and Forecasting*, 7(4):699–709. doi:10.1175/1520-0434(1992)007<0699:CATCRA>2.0.CO;2.

Barnston, A. G. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4):679–688. doi:10.1016/j.ijforecast.2006.03.001.

Baseri, H. and Belali-Owsia, M. (2017). A novel hybrid ica-anfis model for prediction of manufacturing processes performance. *Part E: Journal of Process Mechanical Engineering*, 231(2):181–190. doi:10.1177/0954408915585256.

Betterton, C. E. and Silver, S. J. (1988). Detecting bottlenecks in serial production lines—a focus on interdeparture time variance. *International Journal of Production Research*, 50:4158–4174. doi:10.1080/00207543.2011.5968473.

Bhamu, J. and Sangwan, S. K. (2014). Lean manufacturing: literature review and research issues. *International Journal of Operations & Production Management*, 34(7):876–940. doi:10.1108/IJOPM-08-2012-0315.

Bi, Z., Xu, L. D., and Wang, C. (2014). Internet of things for enterprise systems of modern manufacturing. *IEEE Transactions on Industrial Informatics*, 10(2):1537–1546. doi:10.1109/TII.2014.2300338.

Camps-Valls, G., Bandos Marsheva, T., and Zhou, D. (2007). Semi-supervised graph-based hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 45(10):3044–3054. doi:10.1109/TGRS.2007.895416.

Cao, Z., Deng, J., Liu, M., and Wang, Y. (2012). Bottleneck prediction method based on improved adaptive network-based fuzzy inference system (anfis) in semiconductor manufacturing system. *Chinese Journal of Chemical Engineering*, 20(6):1081–1088. doi:10.1016/S1004-9541(12)60590-4.

Chai, T. and Draxler, R. R. (2014). Root mean square error (rmse) or mean absolute error (mae)? – arguments against avoiding rmse in the literature. *Geoscientific Model Development*, 7(3):1247–1250. doi:10.5194/gmd-7-1247-2014.

Chang, Q., Ni, J., Bandyopadhyay, P., Biller, S., and Xiao, G. (2007). Supervisory factory control based on real-time production feedback. *Journal of Manufacturing Science and Engineering*, 129:653. doi:10.1115/1.2673666.

Ciresan, D., Meier, U., and Schimdhuber, J. (2012). Multi-column deep neural networks for image classification. In *Proceedings of 2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3642–3649. doi:10.1109/CVPR.2012.6248110.

Dawson, C. (2016). *Applied artificial neural network*. MDPI AG, Basel.

Deng, X., Liu, Q., Deng, Y., and Mahadevan, S. (2016). An improved method to construct basic probability assignment based on the confusion matrix for classification problem. *Information Sciences*, 340-341:250–261. doi:10.1016/j.ins.2016.01.033.

Dietterich, T. (1995). Overfitting and undercomputing in machine learning. *ACM Computing Surveys (CSUR)*, 27(3):326–327. doi:10.1145/212094.212114.

Faget, P., Eriksson, U., and Herrmann, F. (2005). Applying discrete event simulation and an automated bottleneck analysis as an aid to detect running production constraints. in m. e. kuhl, n. m. steiger, f. b. armstrong, & j. a. joines (eds.). In *Proceedings of the 2005 Winter Simulation Conference*, page 1401–1407. doi:10.1109/WSC.2005.1574404.

Frazier, G. L., Spekman, R. E., and O'Neal, C. R. (1988). Just-in-time exchange relationships in industrial markets. *Journal of Marketing*, 52(4):52–67. doi:10.2307/1251633.

Gareth, J., Witten, D., Hastie, T., and Tibshirani, R. (2015). *An Introduction to Statistical Learning*. Springer, New York.

Gerven, M. A. J. v. and Bohte, S. M. (2017). Editorial: Artificial neural networks as models of neural information processing. *Frontiers in Computational Neuroscience*, 11. doi:10.3389/fncom.2017.00114.

Goldratt, E. M. (1990). *Theory of Constraints*. North River Press, Croton-on-Hudson, NY.

Gupta, P. (2017). Decision trees in machine learning. https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052, visted: 2018-02-17.

Hopp, W. and Spearman, M. L. (2001). *Factory physics: Foundations of manufacturing management (2.th ed.)*. McGraw Hill, NewYork.

Howley, T., Madden, M. G., O'Connell, M., and Ryder, A. G. (2006). The effect of principal component analysis on machine learning accuracy with high-dimensional spectral data. *Knowledge-Based Systems*, 19(5):363–370. doi:10.1016/j.knosys.2005.11.014.

Jack Hu, S. (2013). Evolving paradigms of manufacturing: From mass production to mass customization and personalization. *Procedia CIRP*, 7:3–8. doi:10.1016/j.procir.2013.05.002.

Jensen, K. (2012). A diagram showing the relationship between the different phases of crisp-dm and illustrates the recursive nature of a data mining project. [Online; accessed April 27, 2018].

Jeschke, S., Brecher, C., Song, H., and Rawat, D. B. (2017;2016). *Industrial internet of things: Cybermanufacturing systems*. Springer International Publishing. doi:10.1007/978-3-319-42559-7.

Konenenko, I. (2001). Machine learning for medical diagnosis: History, state of the art and perspective. *Artificial Intelligence in Medicine*, 23(1):89–109. doi:10.1016/S0933-3657(01)00077-X.

Kotsiantis, S. B., Zaharakis, I. D., and Pintelas, P. E. (2006). Machine learning: A review of classification and combining techniques. *Artificial Intelligence Review*, 26(3):159–190. doi:10.1007/s10462-007-9052-3.

Kuderer, M., Gulati, S., and Burgard, W. (2015). Learning driving styles for autonomous vehicles from demonstration. In *Proceedings of 2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2641–2646. doi:10.1109/ICRA.2015.7139555.

Längkvist, M., Karlsson, L., and Loutfi, A. (2014). A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42(1):11–24. doi:10.1016/j.patrec.2014.01.008.

Lasi, H., Fettke, P., Kemper, H.-G., Feld, T., and Hoffmann, M. (2014). Industry 4.0. *Business & Information Systems Engineering*, 6(4):239–242. doi:10.1007/s12599-014-0334-4.

Lawrence, R. S. and Buss, A. H. (1994). Shifting production bottlenecks: Causes, cures and conundrums. *Cogent Engineering*, 3:21–37. doi:10.1111/j.1937-5956.1994.tb00107.x.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to ´ document recognition. In *Proceedings of the IEEE*, volume 86(11), page 2278–2324.

Li, L., Chang, Q., and Ni, J. (2009). Data driven bottleneck detection of manufacturing systems. *International Journal of Production Research*, 47:5019–5036. doi:10.1080/00207540701881860.

Li, L., Chang, Q., Xiao, G., and Ambani, S. (2011). Throughput bottleneck predic-

tion of manufacturing systems using time series analysis. *Journal of Manufacturing Science and Engineering*, 133(2):1–8. doi:10.1115/1.4003786.

Maimon, O. and Rokach, L. (2010). *Data mining and knowledge discovery handbook (2nd;2; ed.)*. Springer, New York.

Opitz, D. and Maclin, R. (1999). Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198. doi:10.1613/jair.614.

Pandian, A. and Ali, A. (2013). Automotive assembly line production loss prediction based on arma-ann model. In *Proceedings of the IIE Annual Conference*, page 2571.

Roser, C., Nakano, M., and Tanaka, M. (2001). A practical bottleneck detection method. in b. peters, j. smith, d. medeiros, & m. rohrer (eds.). In *Proceedings of the 2001 Winter Simulation Conference*, page 949–953.

Roser, C., Nakano, M., and Tanaka, M. (2002). Shifting bottleneck detection. In *Proceedings of the 2002 Winter Simulation Conference*, page 1079–1086. doi:10.1109/WSC.2002.1166360.

Samattapapong, N. and Afzulpurkar, N. (2016). A production throughput forecasting system in an automated hard disk drive test operation using grnn. *Journal of Industrial Engineering and Management*, 9(2):330–358. doi:10.3926/jiem.1464.

Seal, H. L. (1967). Studies in the history of probability and statistics. xv: The historical development of the gauss linear model. *Biometrika*, 54(1/2):1. doi:10.2307/2333849.

Sengupta, S., Das, K., and VanTil, R. P. (2008). A new method for bottleneck detection. in s. j. mason, r. r. hill, l. m. önch, o. rose, t. jefferson, & j. w. fowler (eds.). In *Proceedings of the 2008 Winter Simulation Conference*, page 1741–1745.

Shanmuganathan, S. and Samarasinghe, S. (2016). *Artificial neural network modelling (1st 2016. ed.)*. Springer International Publishing.

Subramaniyan, M. (2015). Production data analytics—to identify productivity potentials.

Subramaniyan, M., Skoogh, A., Gopalakrishnan, M., and Hanna, A. (2016a). Real-time data-driven average active period method for bottleneck detection. *International Journal of Design & Nature and Ecodynamics*, 11(3):428–437. doi:10.2495/DNE-V11-N3-428-437.

Subramaniyan, M., Skoogh, A., Gopalakrishnan, M., Salomonsson, H., Hanna, A., and Lämkull, D. (2016b). An algorithm for data-driven shifting bottleneck detection. *Cogent Engineering*, 54:1–19. doi:10.2495/DNE-V11-N3-428-437.

Subramaniyan, M., Skoogh, A., Salomonsson, H., Bangalore, P., and Bokrantz, J. (2018). A data-driven algorithm to predict throughput bottlenecks in a production system based on active periods of the machines. *Computers and Industrial Engineering*. doi:10.1016/j.cie.2018.04.024.

Susto, G. A., Schirru, A., Pampuri, S., McLoone, S., and Beghi, A. (2015). Machine learning for predictive maintenance: A multiple classifier approach. *IEEE Transactions on Industrial Informatics*, 11(3):812–820. doi:10.1109/TII.2014.2349359.

Tole, A. A. (2013). Big data challenges. *Database Systems Journal*, (3):31–40.

Wirth, R. and Hipp, J. (2000). Crisp-dm: Towards a standard process modell for data mining. In *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining*, page 29–39.

Witten, I. H., Frank, E., Hall, M. A., and Pal, C. J. (2017). *Data mining: Practical machine learning tools and techniques (4th ed)*. Amsterdam: Morgan Kaufmann, Amsterdam.

Yan, L., Miller, D. J., Mozer, M. C., and Wolniewicz, R. (2001). Improving prediction of customer behavior in nonstationary environments. In *Proceedings of IJCNN '01. International Joint Conference*, volume III, pages 2258–2263. doi:10.1109/IJCNN.2001.938518.

Yan, X. and Su, X. (2009;2014). *Linear regression analysis: Theory and computing*. World Scientific, Hackensack, NJ; Singapore.

Yin, S. and Kaynak, O. (2015). Big data for modern industry: Challenges and trends [point of view]. *Proceedings of the IEEE*, 103(2):143–146. doi:10.1109/JPROC.2015.2388958.

Zhang, L., Zhang, L., Tao, D., and Huang, X. (2012;2011). On combining multiple features for hyperspectral remote sensing image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 50(3):879–893. doi:10.1109/TGRS.2011.2162339.

Zhou, J. (2013). Digitalization and intelligentization of manufacturing industry. , 1(1):1–7. doi:10.1007/s40436-013-0006-5.

Zissis, D., Xidias, E. K., and Lekkas, D. (2015). A cloud based architecture capable of perceiving and predicting multiple vessel behaviour. *Applied Soft Computing*, 35:652–661. doi:10.1016/j.asoc.2015.07.002.