Machine Learning solutions for exception handling

March 2019

Justin Fennis

Master Thesis – Industrial Engineering & Management

Production & Logistics Management

UNIVERSITY OF TWENTE.

Supervisors

Dr. M.C. van der Heijden

Dr. N. Knofius

IBM

Supervisor

Ir. J.P. Hazewinkel

Preface

Here it is: the report of my Master assignment performed at IBM, which puts an end to my student life. *Time flies when you are having fun*. That is exactly how I experienced the past two-and-a-half years. After finishing my Bachelor Thesis in August ‘16, I had no idea what the next step would be. Therefore, it was an easy choice to take a break for a few months and travel with my girlfriend, who took a sabbatical as well, though Asia.

After returning in January ’17, I decided to stay in Enschede and start my Master study *Industrial Engineering & Management* in February. From the beginning, I put more effort in my study, which was noticeable in the grades I got. Next to studying, I was expanding my own business in Almere in the winter, working as a teach assistant at the University in the other months and travelling as much as possible. Due to this busy schedule, it is not a surprise that it feels like yesterday that I started my Master.

After finishing my courses, it was time to find a Master assignment. I was looking for an assignment in *de Randstad* and *Matthieu* offered me an assignment at IBM, Amsterdam. He put me in contact with *Jaap* and a few days later we had an appointment to discuss the assignment. We agreed on the assignment and I started after the summer holiday!

Although I was a newbie to IBM and the *working life*, I felt home right away. Since the assignment was a follow up research, it was sometimes difficult, but *Jaap* was always able to give me ideas and insights to continue. I want to thank him for supervising me during my 6 months at IBM. Next to *Jaap,* I want to thank all the colleagues who helped me during my assignment. *Melle* and *Menno,* thanks for being patient and for helping me. *Hans,* thanks for letting me win so many table tennis games!

Besides from the employees of IBM, I want to thank *Matthieu* and *Nils* for supervising me and providing feedback during my assignment. After every meeting I had the idea I was lost, but the feedback helped me a lot!

Last but not least, I want to thank my parents, girlfriend and friends for their help and support.

Although I am not sure what my next step will be, I am sure my student life was one of the best periods in my life!

Justin Fennis
Amsterdam, March 2018

Executive summary

Problem description

IBM is one of the biggest IT-companies worldwide. One of their activities is providing hardware solutions and offering service contracts to their customers. The Service Parts Operations department within IBM is responsible for keeping enough spare part inventory to fulfil these contracts. Their goal is to do this at minimal costs and to do so, many processes are automated by using the software tool *Servigistics*. When problems occur for which *Servigistics* cannot make an automated decision, the exception handling process is triggered. *Servigistics* categorizes these problems (called exceptions) into different review reasons and informs the planners by sending an alert. In 2018 a total of about 35.000 exceptions were triggered. Currently, the performance of the decisions by the planner is unknown.

In this research, we examined the possibilities of using machine learning techniques to predict whether an action is required on a review reason or not, with the goal to reduce the number of exceptions at which planners should have a look.

Solution approach

Our research is focused on four review reasons, which account for about 40% of all exceptions:

- R24: Projected stock out
- R25: Stocked out
- R26: Below Must Order Point
- R83: Project inventory below Must Order Point

Since *Servigistics* does not allow us to retrieve historical data, data was gathered over a period of one-and-a-half months. The exceptions, the actions taken by the planners and the data on which the decisions have been taken, are gathered for these review reasons. From the original data, 43 characteristics (called features) were derived, which give information about the exception and the part connected to the exception. These features can be categorized as:

- Part characteristics
- Inventory information
- Order information
- Tactical settings
- Exception related

Since the performance of the decisions of the planners is unknown, a performance indicator is introduced which qualifies the decision taken as *good* or *bad*, which revealed that about 65% of the decisions taken are qualified as *good*. This performance indicator is used to create two data sets: one containing all exceptions and one containing only the exceptions qualified

as *good*. Using the first data set for building a model will only result in a more efficient process, whereas the second data set could also make it more effective.

Data exploration revealed that we are dealing with missing data for certain features.

In the data preparation step, missing data is imputed, new features are constructed and the most relevant features are explored. Missing data is imputed by using different techniques. New features are derived from the original data. These features are introduced, since they bring new information about the exception.

Experiments are performed based on several operations which adjust the data. These experiments are shown in Table 1 and are conducted per review reason.

Experiment	**All exceptions / Good decisions**	**Impute missing data?**	**Feature selection?**
1	All	Yes	Yes
2	All	Yes	No
3	All	No	Yes
4	All	No	No
5	Good	Yes	Yes
6	Good	Yes	No
7	Good	No	Yes
8	Good	No	No

Table 1: Experiments overview

For various reasons, we have chosen to use decision tree algorithms as classifier algorithm. The *auto classifier* in SPSS Modeler evaluated 7 decision tree algorithms and C5.0 came out to perform best. The C5.0 algorithm settings are tuned by performing experiments for the settings *minimum number of records per child branch* (2 and 5) and the *pruning severity* (25 and 75).

To evaluate the performance, the datasets are partitioned in a training set of 80% of the data and a testing set of 20% of the data. On the training set, *5-fold cross validation* is performed to determine the performance of the algorithm. The 5 models built during *cross validation* are combined to one ensemble model and the performance is tested by applying it to the testing set.

Results

Out of all these calculations it shows that the best performance is achieved if the algorithm has 2 as value for the *minimum number of records per child node* and 25 as *pruning severity*. Next to that, on average, the performance is higher for datasets which consist of exceptions on which the action taken is qualified as *good.* Imputing missing data manually seems to decrease the performance. Applying feature selection before the modelling step, has barely influence on the performance.

Decision tree algorithms have the most predicting power for review reason R25 (stocked out) and R83 (projected inventory below must order point) in whether an action should be taken on an exception or not. The review reason R24 (projected stock out) is less predictable and R26 (inventory below must order point) is not predictable at all. The limited size of the data sets has a significant impact on the performance of the models, since less extensive decision trees can be built. The decision trees built for R83 are much deeper compared to R24, R25 and R26 and the standard deviation of the results is much lower. Despite the limited size of the data set of R25, the models perform well, possibly because whether an action should be taken or not on a current stock out situation is clearer. The models for R83 perform slightly worse, but more stable, due to the bigger data set compared to the other three review reasons. For both R25 and R83 patterns which make sense are found in the models.

The performance can be further increased by considering only exception for which the prediction confidence level exceeds a certain threshold.

The ensemble models perform worse than the *cross validated* models and on top of that, the results deviate significantly from the *cross validated* results.

Based on these results, we conclude that using machine learning for the exception handling is promising, but the performance should increase to be a true planner assistant. The approach led to a structured and clear research and it is advised to use this approach in future research as well.

Limitations & future research

Since the main limitations were the lack of involvement of actual planners in India and the limited number of records in the dataset, we advise IBM to perform a future research in which the planners are involved in the process, such they can provide valuable features, and gather data for a longer period to increase the size dataset. This could lead to an increase in performance and more stable models. Involving the planners would also replace the performance indicator for the decisions, which is currently only an approximation of the quality of decisions.

Furthermore, we recommend considering online learning algorithms as a next step, in which the exceptions, their circumstances and the action taken by planners are tracked automatically and the model is updated after every new exception. An advantage of such an algorithm is the possibility of incorporating the reasoning of a decision by a planner.

Abbreviations

AUC	–	Area Under the Curve
EMEA	–	Europe, Middle East and Africa
IBM	–	International Business Machines Corporation
ML	–	Machine Learning
MOP	–	Must Order Point
SPO	–	Service Parts Operations

x

UNIVERSITY
OF TWENTE.

List of figures

List of tables

Table of contents

Chapter 1: Introduction

For completing my Master Industrial Engineering & Management at the University of Twente, a research is performed within IBM. This research is a follow-up research on a prior research by Schultz (2017), where the possibilities of using machine learning techniques within the exception handling processes at the Service Parts Operations department within IBM are examined. In this chapter, first an introduction to IBM and the Service Parts Organization is given and *machine learning* is introduced. Next, the research design is explained and finally the structure of the report will be given.

1.1 IBM

IBM is one of the biggest IT-companies worldwide with more than 400.000 employees. IBM was incorporated in 1911 as the *Computing Tabulating Recording Company*, although its origins can be traced back to the end of the 19th century. In 1924 the company was renamed to IBM. Although IBM was a big player on the consumer electronic products in the 20th century, nowadays the main focus of IBM is inventing technology and applying it to business and society on a global scale to make the world work better (IBM, 2018).

Spare Parts Management

One of the activities of IBM is providing hardware solutions to its customers and offering service contracts to maintain the hardware at the customer site. Two reasons why customers buy products of IBM are their reliability and performance (IBM, 2014). Spare Parts Management (SPM) plays an important role for these activities. According to Moncrief et al. (2006) SPM contains all activities in order to have exactly in stock what is needed, when it is needed. The objective of SPM is to keep enough inventory to be able to deliver a certain predefined service level, but at minimal costs.

Figure 1: SPO organizations around the world (Koopman, 2011)

Within IBM, the SPO organization is responsible for SPM. IBM has a dense logistic network that can provide spare parts to their customers within up to two hours. The last decade IBM has focused on globalization of its processes, which led to the change of key processes in SPO. Next to that, IBM invested money in improving the planning of spare parts. Four main SPO departments are located around the world (Figure 1): EMEA (yellow), Asia Pacific (green), United States (blue) and Latin America (red) (Koopman, 2011). The EMEA-region is the focus of this research. The head quarter of the SPO organization for the EMEA-region (Europe, Middle-East and Africa) is located in Amsterdam and the central inventory buffer for the EMEA-region is located in Venlo, which provides spare parts for the local warehouses within the EMEA-region (Figure 2).

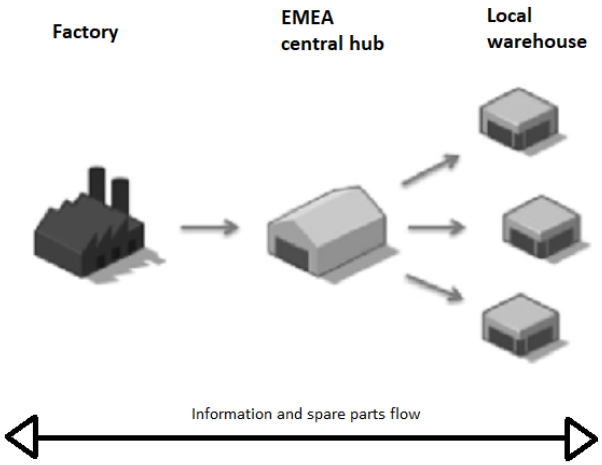

Figure 2: Spare parts supply chain (Cohen et al., 2006)

The SPO department uses two systems to regulate their Spare Parts Management, CPPS and *Servigistics*. CPPS is the database where real time information of spare parts is stored. To be able to deliver a certain service level, *Servigistics* uses information on spare parts from CPPS and makes predictions of demand at the central buffer for different spare parts every day. Based on the predicted demand and current spare part information, *Servigistics* automatically places orders to keep enough inventory at the central buffer. Although these systems keep getting better and better and therefore this process is highly automated, different problems can arise, where a planner of the SPO department should have a look at. When the systems encounter something unusual, the exception handling process is triggered and *Servigistics* sends an alert to its planners. These exceptions can be of different reasons, so called review reasons. In total, there are 81 different review reasons. Some of the review reasons are only informative, but some can be problematic, for example a situation where not enough stock will be available in the entire EMEA-region for a certain part. In these situations, the planner examines the problem and takes and action if needed. Although the SPO department of the EMEA-region is located in Amsterdam, the planners (which make decisions or give advice on the decisions) and the team leaders (which approve, decline or adjust the decisions made by the planners) are located in India.

1.2 Machine learning

Machine learning (ML) is a methodology that uses statistical techniques and algorithms to be able to learn from data and identify patterns in data. ML is getting more and more attention and according to Jha (2017) it is expected that ML algorithms will replace a large amount of jobs worldwide. The definition of ML according to IBM is as follows:

Machine learning is the science that examines how computers gain knowledge from data, using algorithms and analysis models.

One of the learning techniques of ML is supervised learning, where a model is given a historical dataset of inputs and corresponding outputs. With this data, classification algorithms will try to find hidden patterns in the data and learn from that. When enough data is fed to an algorithm and the data is of high quality, the model could be able to predict the outcome for new situations. These tools become more and more accurate and are

approaching human intelligence. For this reason, machine learning becomes a standard in companies and is often used as supportive tool to gain competitive advantage (IBM, 2018).

1.3 CRISP-DM

In literature several methodologies for conducting a structured and successful machine learning research are known. Cross Industrial Standard Process for Data Mining is one of them and is developed by Daimler Chrysler, SPSS and NCR. It provides a uniform framework and guidelines for data miners and is well accepted due to its complete and well documented approach (Santos & Azevedo, 2005) compared to other methodologies.

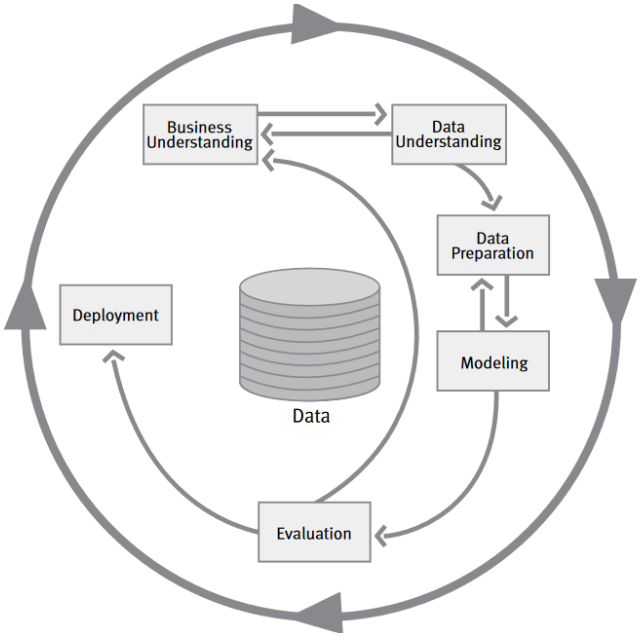

Figure 3: Steps of the CRISP-DM methodology (Chapman et al., 2000)

The CRISP-DM methodology is visible in Figure 3 and consists of six steps:

1. **Business understanding**: the first step focusses on getting familiar with the company and understand the problem. This step leads to a project plan.
2. **Data understanding**: in the second step, the data to collect is defined, initial data is gathered and is being evaluated to get insights into the data.
3. **Data preparation**: in the third step, the final dataset(s) is (are) constructed from raw data. This includes activities like handling missing data, transforming data, feature extraction, feature selection and cleaning.
4. **Modelling**: in this step, modelling techniques are chosen, parameters are set and modelling takes place.
5. **Evaluation**: after the modelling, the results are evaluated before going to the deployment step.
6. **Deployment**: in the final step, the knowledge gained from the research is put into practice such that it can be used.

These steps and the tasks per step are visible in Figure 4. This figure is a slightly adapted version of the one of Chapman et al. (2000) and does not particularly follow a certain order. It is allowed to go back and forth between the different steps.

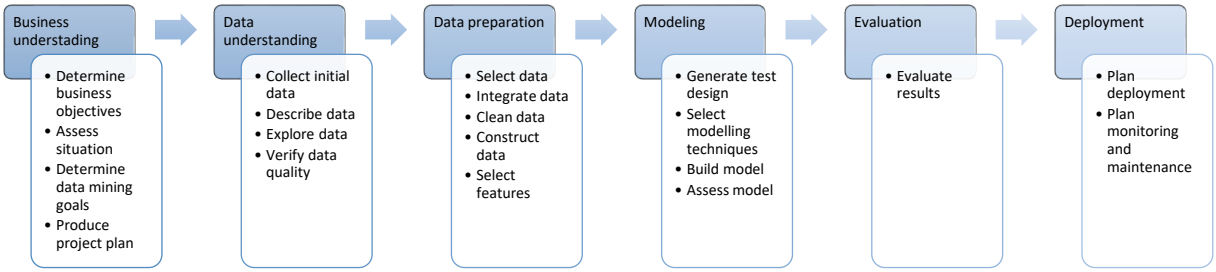

Figure 4: Tasks per step in CRISP-DM (Chapman et al., 2000)

1.4 Research design

In this section the research design is explained. First the motivation of the research is explained, next the problem statement is given with its goal and the research questions are stated.

Motivation

Currently, the planners of the SPO organization in the EMEA-region deal with making decisions on about 35,000 exceptions triggered by *Servigistics* for the central buffer on a yearly basis, which requires significant time. *Servigistics* provides information of the part to the planners on which a decision should be made. This information could be about the price of the part, current inventory position of the EMEA-region or local warehouses, historical and predicted demand, open orders for the EMEA-region and so on. Based on this information, a planner makes a decision and in some cases a team leader should give an approval on the decision made. The center of competence for the exception handling process is located in India. However, planners in India change job frequently, which means that planners at IBM are often inexperienced. Next to that, currently neither the consistency of these decisions nor the quality is known.

Based on these problems, IBM wants to examine the possibility of using machine learning in their exception handling process to support their planners and reduce the effort planners put in the exception handling process, by removing the exceptions on which no action is required.

In a prior research (Schultz, 2017), the possibility of using machine learning techniques for review reason R38 is examined. R38 is one of the many review reasons within the exception handling process and is triggered when the recommended orders result in an order increase outside lead-time, which means that *existing orders would result in the inventory level dropping below the safety stock in a four-day period after the current date plus the lead-time*. Supervised machine learning was applied to examine whether it was possible to reproduce the planner's decisions. However, the derived model was only able to predict a planner's decision in about 59% of the cases. The low performance was explained by the low data quality. Even though, since the planner's decision quality is unknown, the chosen performance measure only offers limited explanatory power. Schultz (2017) proposed a performance

indicator for the quality of decisions, however the performance was not included in the model. Finally, Schultz (2017) did not follow a structured approach to create a proper learn set, which puts the quality of the learn set at doubt.

Research problem

As concluded in the previous section, the problem is whether the learn set used for machine learning in Schultz' research (2017) was good enough to learn from. This problem and the willingness of IBM to examine the possibilities of using machine learning techniques in the exception handling process to reduce the planner's effort put into the exception handling process, by removing the exceptions on which no action is required, led to the following research problem:

How can machine learning techniques be effectively applied on different review reasons to improve the efficiency and effectiveness of the exception handling process?

Research goal

In this research we evaluate the possibility to create a data set suitable for supervised machine learning. Furthermore, if the data set provides sufficient predictive power, we will build a demonstrator to exemplify the possibility to use machine learning techniques during the exception handling process.

Research questions

To conduct a proper research and get an answer on the research problem, it is chosen to follow the CRISP-DM methodology. The research problem is divided into research questions, which belong to the different steps of the CRISP-DM methodology.

Business understanding & data understanding

This chapter, *Chapter 1: Introduction*, belongs to the first step of the CRISP-DM methodology. However, to get a better understanding of the problem, it should be known what the SPO department is and how their processes take place. Furthermore, review reasons are selected and the initial data is evaluated.

Research question 1: *What is the current situation regarding the exception handling process at the SPO department?*

- *What is Service Parts Management?*
- *What does the exception handling process look like?*
- *What is the current efficiency of the exception handling process?*
- *What review reasons are selected to focus on?*

- *On what information are decisions taken and what data is available in the systems of IBM and must be gathered?*
- *What patterns can already be found in the initial data?*
- *How can the decisions made by the planners be made measurable?*

Literature review

After studying the problem setting, we examine the literature to gain insights about how to create a learn set and what modelling techniques may be applicable.

Research question 2a: *How can a learn set be created?*

- *How to deal with missing values?*
- *What feature selection methods are available?*

Research question 2b: *What modelling techniques can be used?*

- *Which algorithms are suitable to use?*
- *What performance measures should be used?*

Data preparation

In this step, the initial data is processed into a final dataset which can be used for the modelling step.

Research question 3: *How can a proper learn set be created for the selected review reasons?*

- *How should the data be prepared to derive a problem specific learn set?*
- *What features are selected?*

Modelling

From the previous step, based on the decision performance metric, different learn sets are available. Using these learn sets, we select and apply a suitable machine learning technique.

Research question 4: *What does the machine learning model look like?*

- *What machine learning techniques are being used and under what settings?*

Evaluation

When machine learning is applied, the results should be evaluated.

Research question 5a: *What are the performances of the machine learning models?*

- *How do the machine learning models perform per learn set?*

Research question 5b: *Under which condition(s) is it (not) possible to apply machine learning during the exception handling process and why (not)?*

- *Which review reasons can be automated such that they can have a beneficial effect on the efficiency of the exception order handling process at the SPO department?*
    - *For well-performing review reasons: what are the model characteristics?*

Deployment

The final step of the CRISP-DM methodology is to make the results operational.

Research question 6: *What would be the next steps to implement a ML-solution?*

After the CRISP-DM methodology is finished, a final chapter with the conclusion and recommendations will be presented.

Scope

In order to conduct an efficient and effective research, the scope of this research should be clearly demarcated. This is due to the complexity of this research and the time constraints.

The exception handling process is the focus of this research. This process handles all exceptions for which *Servigistics* cannot make an automated decision. *Servigistics* categorizes these problems into different review reasons. Since it is not possible to have a look at all review reasons and not all review reasons are suitable to apply machine learning on, a selection will be made. For the selection, we consider the review reason frequency, the possibility to automate associated actions, and the review reason's importance. By focusing on selected review reasons only, we are able to demonstrate the key steps of our approach. If the results of this research are promising, the same approach can be applied to other review reasons.

Since IBM is the owner and developer of SPSS Modeler, IBM expressed a preference to use SPSS Modeler 18.1 for the execution of this research. SPSS Modeler is a data mining tool, which supports the development of predictive models. It contains several data exploration options and machine learning algorithms such as decision trees and statistical learning methods.

1.5 Structure of the report

The chapters and corresponding research questions are summarized in Table 2.

Chapter	Research question(s)	Data sources
2: Business- and data understanding	*RQ 1*	IBM employees IBM documentation IBM databases Literature
3: Literature review	*RQ 2*	Literature IBM documentations IBM employees
4: Data preparation	*RQ 3*	Interviews with IBM employees IBM databases
5: Modelling	*RQ 4*	Prior research (Schultz, 2017) IBM databases IBM SPSS modeler (YouTube) tutorials
6: Evaluation	*RQ 5*	IBM SPSS modeler
7: Deployment	*RQ 6*	
Conclusion, limitations and future research		
References		
Appendices		

Table 2: Chapters with their corresponding research questions and data sources

Chapter 2: Business- and data understanding

In this chapter, the current processes within the SPO department of the EMEA region at IBM will be explained. To do so, first a general introduction to the SPO department will be given, the software being used will be introduced and the current exception handling process is explained. The first two steps of the CRISP-DM methodology are the focus of this chapter, as highlighted in Figure 5.

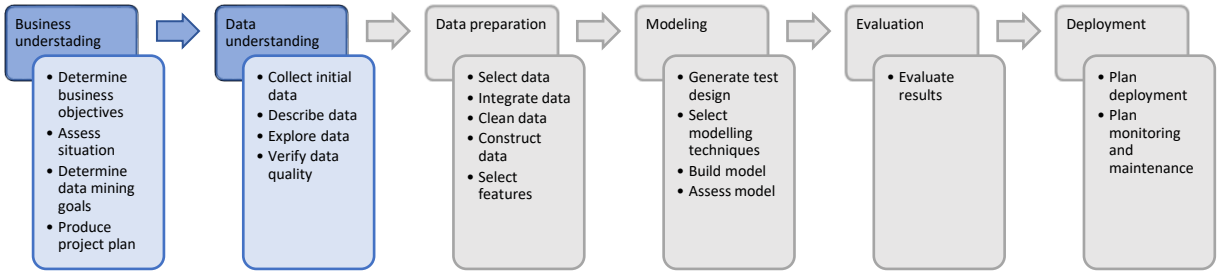

Figure 5: Current focus within CRISP-DM methodology (Chapman et al., 2000)

2.1 SPO department

The SPO (Service Parts Operations) department is responsible for the Service Parts Management of IBM. As already mentioned, the goal of the SPO department is to *provide the right service parts required for maintenance and repairs on IBM hardware at the right time and the right place at minimal costs.* To be able to do so, a balance should be found between availability of service parts and inventory costs (Figure 6). The SPO department in Amsterdam is responsible for the EMEA-region. Their mission is *to manage and/or provide service parts solutions and logistic consultancy to EMEA clients* accompanied to their vision of *being the Service Parts Logistics Competence Centre in EMEA*.

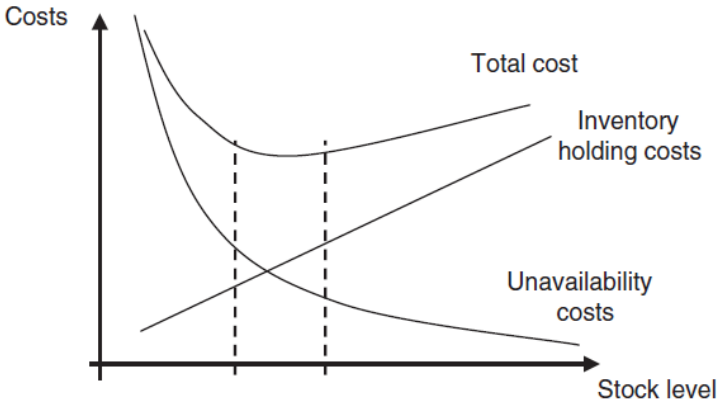

Figure 6: Optimal inventory level (Cavalieri et al., 2014)

To follow their mission, the SPO department has the following major processes:

- **Service Data & Parts Data Management**: Assure that all relevant information on service requirements and parts is timely collected and accurate
- **Parts Planning & Inventory Management**: Create and maintain a stocking plan that fulfills the service requirements at minimal overall cost
- **Parts Ordering & Stocking Management**: Order parts in line with the stocking plan and manage these assets in secure stock locations
- **Parts Allocation & Delivery Management**: Allocate the requested part to a storage location and deliver the part from that location to the right place, at the right time

- **Parts Reverse & Reutilization Management**: Analyze and execute options to reuse parts or dispose
- **Measurements & Controls**: Administration of transactions, measurement of key performances, reporting, financial management, business controls

In the EMEA-region, IBM has more than one million stock keeping units, about 270 stock locations in 63 countries, and service goals for the delivery time at the customer site from two business days down to two hours. This results in a highly complex supply chain, which is shown in Figure 7. The central warehouse plays an important role in the supply chain, since it functions as buffer between the local warehouses and the spare parts suppliers. To be able to act according to the service contracts, the SPO department monitors and controls the availability of parts at the central and local warehouses.

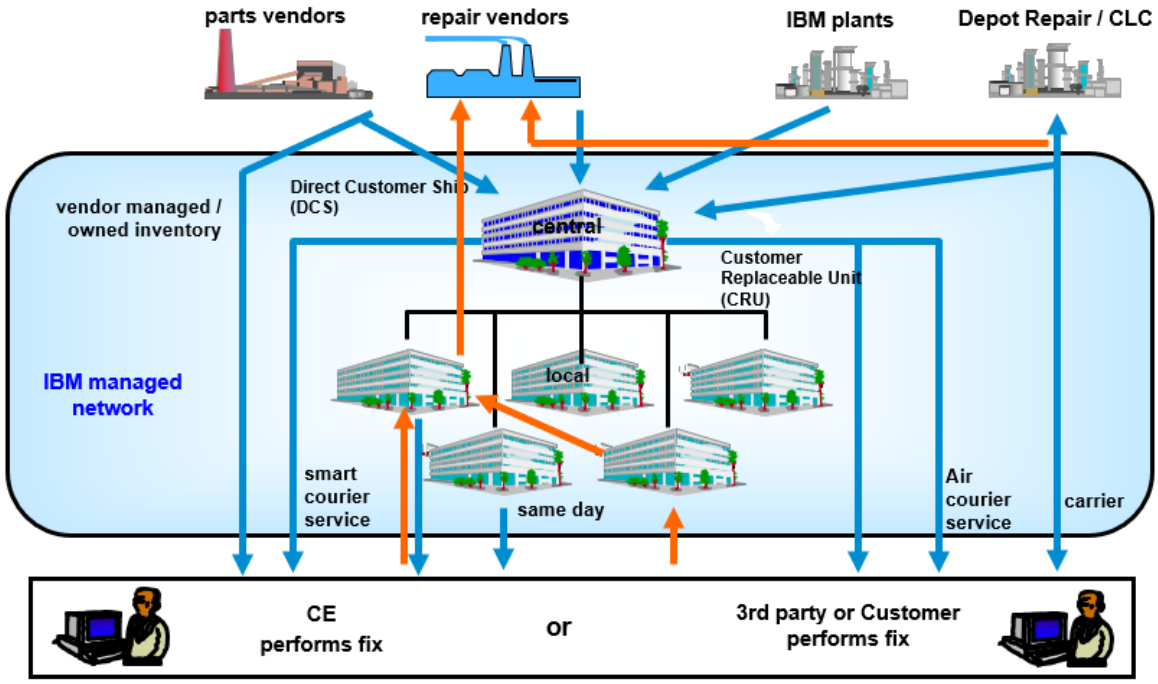

Figure 7: Supply chain of the SPO department (IBM, 2005)

2.1.1 Software being used

In this paragraph the software being used by the SPO department in the EMEA region to optimize the inventory at the central buffer are explained. The main software being used are *Servigistics (*formerly known as *XelusPlan)* and *CPPS*.

Servigistics

Servigistics is the software tool that is used within IBM for control and planning purposes. These activities are on the tactical and operational level. *According* to their user manual (Xelus, 2006), *Servigistics* "ensures that the right part is in the right place to maximize service at minimum costs". While the software is highly automated, it is possible to define customized

exceptions, which require a manual assessment by planners. For IBM, *Servigistics* provides a crucial link between customers and suppliers, since lots of work is automated.

Servigistics is able to predict demand on multiple levels in the supply chain and identifies the amount to order for different stocking locations. For these forecasts and orders, it uses information of other systems and is based on several factors, e.g. historical demand, known trends, quantities on hand, availability of sources, lead times and their requirements, minimum order quantities, projected orders and more. All this item information is visible in the Planner Worksheet. The Planner Worksheet provides its users a comprehensive view of item information in graph- and spreadsheet format. Since it is quite time consuming to plan thousands of parts, most of the planning decisions are automated, unless a certain condition is met. Review reasons are examples of such conditions. The Planner worksheet helps the planners making well considered decisions on the review reasons.

The purpose of the Planner Worksheet is to bring supply and demand together over time. The decisions made over time can be linked to a plan, from which planning rules are established. The Planner Worksheet tracks the result continuously and it gives notifications to the users if predefined thresholds are violated. In Figure 8 the Planner Worksheet is shown. At the left side, the characteristics of the parts are visible and in the middle the inventory over time is visualized with exact inventory levels below the visualization. The vertical black line represents the current date, after which the projections are shown. The other vertical lines give the lead time of the different kind of suppliers. In the lower left corner, the current review reasons are shown and at the lower right corner the notepad can be seen.

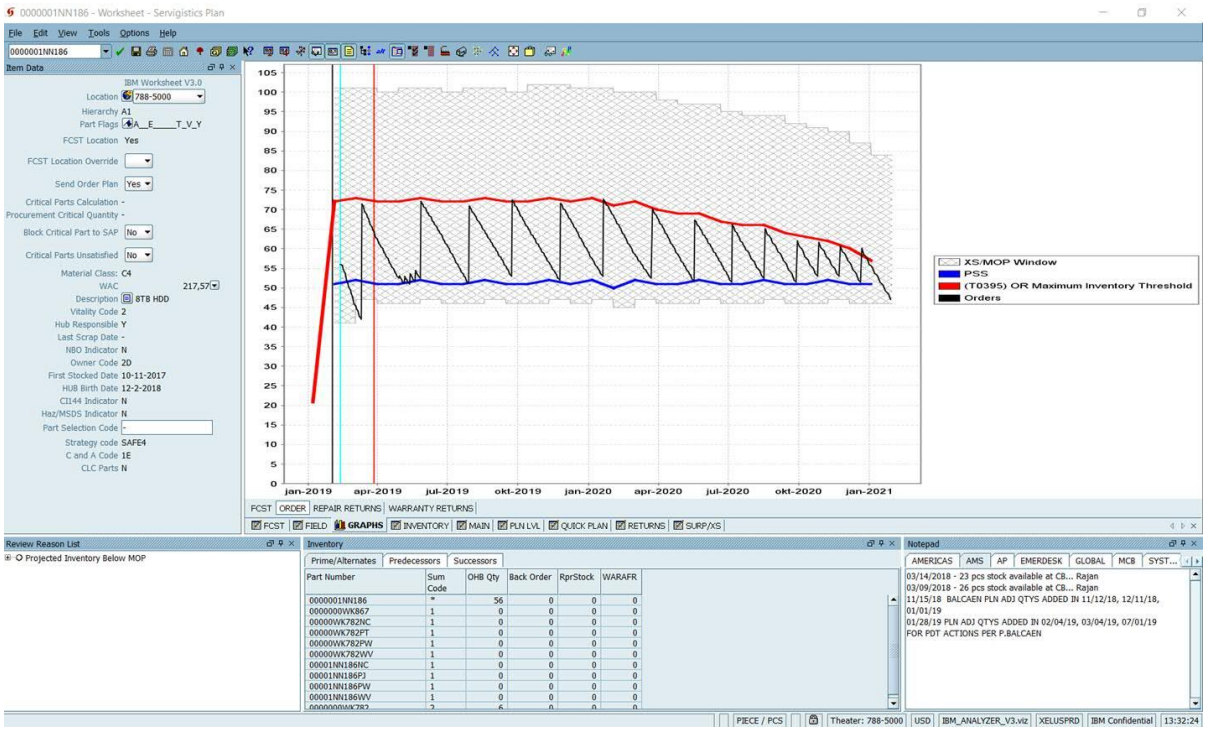

Figure 8: Planner Worksheet

CPPS

All real-time data on inventory at warehouses, and inventory in the pipelines between warehouses is visible in the Common Parts Processes and Services system (CPPS). CPPS is linked to different tools used by IBM and *Servigistics* extracts data of CPPS. Also, data on the decisions made during the exception handling process are extracted from the CPPS database.

2.1.2 Exception handling

Under ideal circumstances orders are placed automatically by *Servigistics* and exactly the amount of inventory is in stock at the central buffer or available in the network to satisfy the service level constraints for each customer. However, problems can arise for which manual interventions are required. These interventions are referred to as exceptions and are triggered by *Servigistics* because one or more predefined conditions, called review reasons, were violated. Examples of review reasons are an expected stock out or an existing backorder quantity. One exception could consist of multiple review reasons. The review reasons indicate what the problem is.

At the competence center in India, about 15 planners work on exceptions every day. Every planner is responsible for a set of parts. Within the Planner Worksheet a so-called Work Queue is displayed that informs the planner which parts require a manual assessment. The items in the Work Queue are prioritized on the review reason's priority. If more than one review reason is linked to a part, only the review reason with the highest priority is displayed. If one of the parts is selected, the Planner Worksheet supports the planner by displaying information about the selected part. Based on the review reasons linked to the part, the planners are expected to take the following steps according to the *Servigistics Manual (Xelus, 2006)*:

- Check network inventory, open orders and future (planned) orders
- Review the forecast for the part
- Check historical demand

Based on the information available on the part, the planner decides what to do. If a decision is taken and the associated review reasons were solved, the part is automatically removed from the Work Queue. However, if any associated review reason remains unsolved, the planner may approve the review reason, which means that the planner had a look at the problem, but the planner cannot resolve the condition which triggers the review reason at the moment. When approving, the review reason is disables for a certain period. This period is review reason specific and is called the *disabling days*.

2.2 Review reasons

Currently there are 80 different review reasons defined in *Servigistics*. From these 80 review reasons, 25 are active and trigger exceptions.

To get more insight in the occurrence of the review reasons, data on the exceptions was gathered from the review reason history log for a period of one year, from 01-09-2017 until 31-08-2018. In total 21 of the 25 active review reasons appeared in the review reason log. In one year, a total of 39,701 review reasons occurred. Since an exception can have more than one review reason, the total number of exceptions is lower: 34,474. Handling of this amount of exceptions requires significant time.

In Appendix A, a table is given with an overview of the twenty-one active review reasons with their occurrence and priority. The table shows that only a few review reasons lead to most of the exceptions. The Pareto-chart in Figure 9 shows this effect. Although the Pareto-rule does not apply, about 33% of the review reasons (R38, R83, R80, R81, R24, R25 and R26) trigger about 80% of the exceptions.

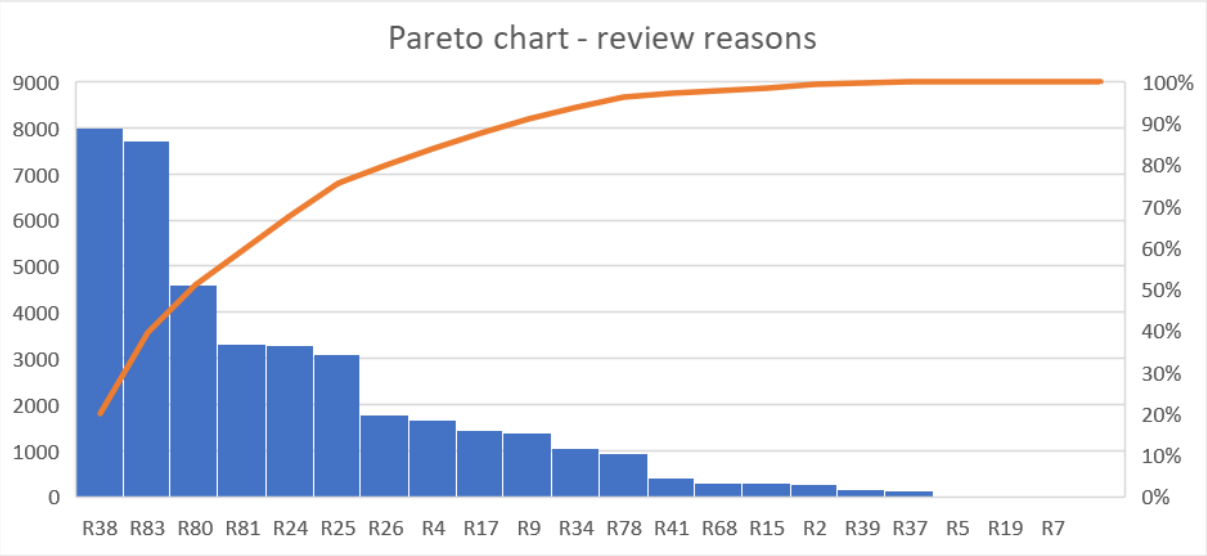

Figure 9: Pareto-chart of the review reasons

2.2.1 Selected review reasons to focus on

In this research, it is chosen to focus on R83, R24, R25 and R26. This decision has been made based on the occurrence, the predefined priority of these review reasons and conversations with the team leaders of the exception handling process. On top of that, these review reasons are interrelated, which makes it possible to evaluate these four review reasons together. Together, these four review reasons account for 40% of the review reasons per year.

Reasons why it was decided against the analysis of R38, R80 and R81 are listed in Table 3. Reasons for not focusing on other review reasons are mainly related to the low occurrence of the review reasons and limited data availability.

Review reason	Reasons not to focus
R38: Order increase outside lead-time	This review reason was the focus of Schultz' (2017) research and no useful results were found. Next to that, from conversations we know that for this review reason always an order should be placed and based on availability a certain order type should be used.
R80: Projected inventory above XS point	Possible actions are limited for this review reason. Often a planner cannot cancel an order, since penalties are high. Options are to scrap an order or balance the order over the world. On top of that, this review reason occurs much less frequent than R83.
R81: Supply constraint overconsumed	The predefined priority of this review reason is lower than the priority of R83. Next to that occurrence is much lower as well.

Table 3: Reasons not to choose R38, R80 and R81

The chosen review reasons are summarized in Table 4 and are explained in Appendix B. These review reasons are similar in the way that they occur when the network stock drops below or will drop below a certain value in the future.

Review reason	Timing	Inventory level
R24: Projected stock out	Within two years	-1
R25: Stocked out	Now	-1
R26: Below MOP	Now	Below MOP
R83: Projected inventory below MOP	Within two years	Below MOP

Table 4: Chosen review reasons

To explain the definition of the chosen review reasons, in Figure 10 an example is given of the inventory level of a part over time (black line). In this figure, the vertical black line represents today. The blue line represents the *policy safety stock (PSS)* and is used to anticipate on variations in usage. The red line represents the *maximum stock level* and is used to prevent ordering too many parts. In a healthy situation, the inventory level moves between the *PSS* and the *maximum stock level*. Next to the *PSS* and the *maximum stock level*, a gray area can be seen in Figure 10. The lower boundary of the gray surface represents the *must order point (MOP)*, which is dependent on the *PSS* and the planning horizon. The upper boundary of the gray area represents the *excess stock level*. The gray area becomes narrower when the planning horizon is further in the future.

R24 is triggered when the inventory level drops to 0 somewhere in the future. If the current inventory level is 0, R25 is triggered. R26 is triggered when the current inventory level drops below the *MOP*, whereas R83 is triggered when the inventory level drops below the *MOP* somewhere in the future.

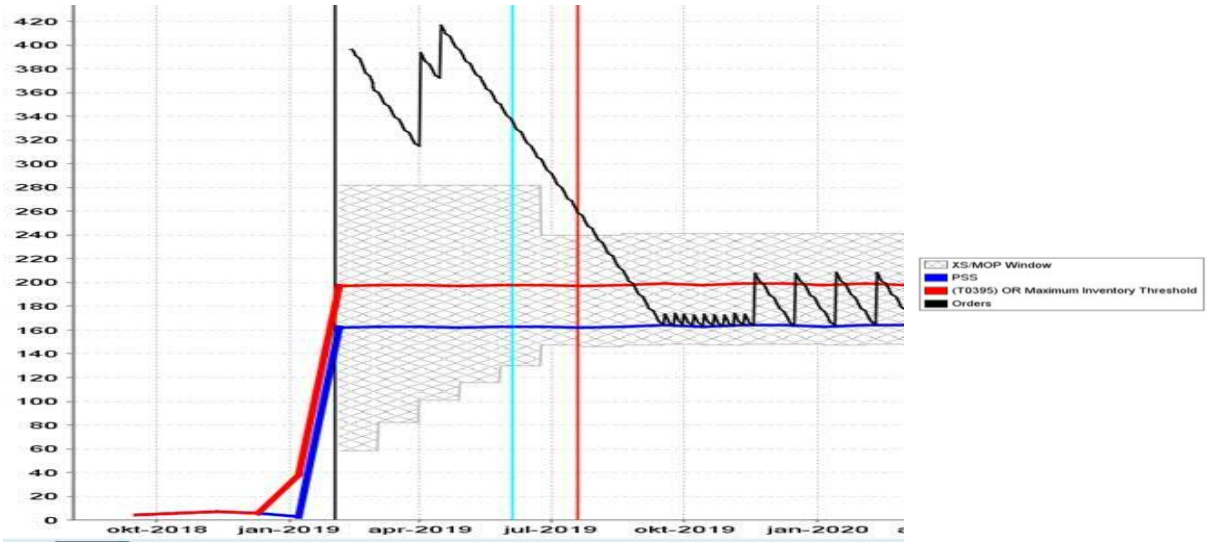

Figure 10: Inventory example

The review reasons may be a consequence of different factors such as missing suppliers, certain parameter settings (e.g. auto-order disabled) or other conditions that prevent *Servigistics* from auto-ordering. Based on the information provided in the Planners Worksheet of *Servigistics* (cf. Section 2.1.1), the planner may take an action to address the factors that have caused the review reason.

2.3 Data understanding

To get a better understanding of the problem, we analyze related data in this section. First, an initial set of data is gathered and we provide an overview of the different data types. Next, the data is explored for patterns and the quality and cleanse is discussed.

2.3.1 Initial data collection

Data to be collected can be divided into the exceptions which are triggered, the features which have influence on the decision and the decision itself.

Exceptions

The exceptions are gathered from the review reason history log. In this log, all review reasons that are triggered by *Servigistics* are listed. Part of this table is shown in Figure 11. The most important information this table holds, is the name of the triggered review reason, the part number on which the review reason was triggered, the date on which the review reason was first entered in the planner's Work Queue, and the date on which the review reason was resolved.

If a review reason is approved (cf. Section 2.1.2), the approval date is given as well.

PartNr_RR	REVIEW_RSN_ID	DISABLING_DAYS	PLN_TURNAROUN_TIME	WORK_QUEUE_DATE	ACTIV_DATE	RVRSN_APPROV_DATE	RVRSN_RESOLV_DATE	LAST_WRKQUEUE_DATE
0000000AE053	R83	10	7	2018-11-10	2018-12-12	2018-11-28	$null$	2018-11-27
0000000AK188	R26	20	7	2018-11-10	2018-12-10	2018-11-09	2018-11-12	2018-11-10

Figure 11: Review reason history log

Features

From interviews with team leaders within the SPO department, factors on which decisions are taken are evaluated. These factors are mainly the information which is provided to the planners by *Servigistics*. Next to that, the tablebook of *Servigistics*, which explains all tables and data which is gathered, is evaluated to find interesting data which can be used. These factors are gathered from the databases of IBM and used as initial input for the machine learning model. Since the planners base their decision on the information shown in *Servigistics*, it is chosen to extract the data from the *Servigistics* databases, although this data is only updated every 7 days. Since *Servigistics* does not allow the retrieval of historical data, we need to record data over a period of time to generate a data set. Due to the limited time of the graduation assignment, the data recording period is limited to the 7th of November until the 21st of December. This is a period of seven weeks and will lead to a dataset of about 3.000 records.

The gathered data can be divided into the following categories:

- Part characteristics, like the weighted average cost, birth date, end-of-service date, material class and shelf life.
- Inventory information, like current stock level, open orders, historical demand and forecasted demand.
- Order information, like supplier availability and lead time.
- Settings, like tactical levels, auto-order indicator, last-time-buy indicator and many other indicators.
- Exception related: number of review reasons for the exception, whether it is triggered in the weekend or not.

This gathered data is directly and indirectly used as features for the exceptions. In Appendix C an overview of the features is given with an explanation. All data of the features are gathered weekly, except for inventory level data, which is gathered daily. The queries, which are used to retrieve the data of features from the databases are provided as supplementary material in the zip-file “MSc Justin Fennis – Queries” and can be found in text in Appendix D. A data model is made in SPSS Modeler to process the data. This model is explained in Appendix E.

Planner decisions

The manual adjustments log, which tracks adjustments the planners make in India, is used as reference for the made decisions. This log shows the following categories of actions:

- Orders and order changes
- Changes in forecast settings
- Settings for auto-order and critical parts
- Changes in schedule values (on which *Servigistics* does its calculations).

In this research, two classes are taken into consideration: take action or do nothing. This is done to reduce complexity, since in the research by Schultz (2017), many decisions were possible, which possibly led to a too complex problem and bad performance of the model. To know the initial decision of the planner, manual adjustments performed in the period of the review reason trigger date until the approval date are considered. However, since a review reason can be approved multiple times and only the most recent approval date is visible, if the approval time is more than the number of disable days after the review reason trigger date, a period of 7 days is taken. The 7 days time window follows from the guideline that a planner should handle their exceptions within 7 days.

Although all adjustments made in *Servigistics* are tracked in the manual adjustment log, it is possible that a planner does not take action within the *Servigistics* environment (and thus no action is tracked in the log), but he decides to take action outside the *Servigistics* environment. Examples could be that a planner triggered an external process, or he decided to communicate over e-mail.

2.3.2 Data exploration

In this section, we explore the data gained from the review reason history log and the adjustments log. First, we show the number of exceptions per review reason, in the time period between 06-11-18 and 13-11-18 in Figure 12. Most exceptions are triggered for review reason 83 (R83). This can be explained by the fact that R83 (projected inventory below must order point) will occur more often than a (projected) stock out, since the must order point is typically larger than -1. For the same reason, a stock out situation (R25) occurs more often than a below must order point situation (R26). One could argue that if a part is out of stock, it is also below the must order point, which induces that there should be at least as much R26 exceptions as R25 exceptions. However, only for planned parts a must order point is defined. Hence, we observe that R25 occurs more often than R26.

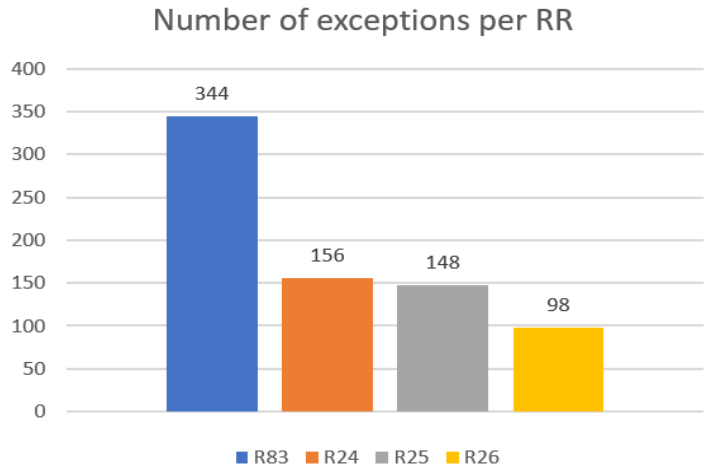

Figure 12: Number of exceptions per review reason

When combining data from the review reason history log with the adjustments log, we can determine the taken actions per exceptions. The chosen approach to combine both data sets is explained in Appendix D. The results are summarized in Table 5. We observe that the majority of the actions for R24, R26 and R83 is a *manual adjustment*. However, on R25 often no action is taken. This seems odd, but this can be explained by the fact that often before a stock-out occurs, other review reasons have already been triggered and actions have already been taken, or that no action can be taken at all.

Review reason	% no action	% action	Action - categories			
			Order-related	*Forecast-related*	*Setting-related*	*Other*
R24	42.3	57.7	*40.0*	*12.2*	*11.1*	*46.7*
R25	63.5	36.5	*31.5*	*11.1*	*16.7*	*57.4*
R26	46.9	53.1	*63.5*	*1.9*	*21.2*	*32.7*
R83	43.3	56.7	*49.2*	*10.8*	*15.4*	*39.5*

Table 5: Distribution action/no action

Overall, we can state that whether a manual adjustment is performed or not is quite balanced, which means that for both the classes *action* and *no action* are quite evenly represented in the dataset. If actions have been taken, we see that most actions are *order related* or fall in the category *other adjustment*. Only limited *forecast related* and *setting related* adjustments are done. This holds for all review reasons.

Data quality

Since a prediction algorithm relies on data, the quality is an important aspect. The data which is used, is directly extracted from the *Servigistics* database. It can be expected that the data in the system is correct and therefore is of high quality. As a check, the values of some records have been checked in CPPS and it came out that values were the same. The *data audit node* of SPSS Modeler performs a quality check, which can be seen in Appendix F. From this quality check we observe that 67.44% of the features have a value for all records. This means that for about one third of the features, there are missing values. A total of 14 of the 43 features contain missing values. Most features contain missing values in 0% - 15% of the records. Four features show more missing values. The feature with most missing values is the *shelflife*. Only 0.94% of the records are complete. Although it is clear that records with a missing value for the feature *shelflife* do not have a shelf life, it cannot be simply imputed by a certain value as is done for other features, since one would expect an unlimited shelf life. We address this issue in Section 4.2, after we examined the literature for different data imputation techniques in Section 3.1.1.

Characteristics of exceptions

When comparing the characteristics of exceptions that were either processed with a manual adjustment or with no action, some insights became clear. As can be seen in Figure 13, when

a part is not stocked at the hub or the hub is not responsible for the part, often nothing is done. This makes sense, but we would expect that no adjustments are performed at all for parts for which the hub is not responsible or for which no inventory is kept.

For exception on which no action has been taken, the weighted average cost, which is the average value of a part for IBM, is lower and higher for exceptions on which an action has been taken. This could indicate that for more expensive parts no action is taken directly and thus more risk is taken.

Finally, often an action is performed for parts with a relative high demand over the last period. This indicates that often action is taken on exceptions for fast-movers and no action is taken on slow-movers. These statistics confirm that different characteristics have influence on the decision which is taken.

Figure 13: Statistics of feature data

Performance of decisions

Not all decisions by the planners are good decisions. Currently, the SPO department does not evaluate the performance of the decisions. Since classification algorithms can only learn from good decisions, the decisions made should get a performance measure. In the stochastic environment of the decisions at the SPO department, it is hard to judge about the decision quality. Although a qualitative measure like approval by team leaders is a suitable technique, a quantitative method is preferable since it is less time consuming. In accordance with team leaders it is chosen to use a simple, though obvious performance evaluator, which evaluates whether the initial decision resolved the review reason. Since different situations are possible, the following assessment metrics are used:

- Based on the decision of the planner, the review reason is resolved directly and the planner does not need to approve the review reason. The decision is qualified as a good one. → Good decision
- Based on the decision of the planner, the review reason cannot be resolved directly. The planner had a look at the review reason, did his best to solve the review reason and approves the review reason with the possibility that the decision made will lead to resolving the review reason in the short future. The following metrics are used for this situation:
    - The decision was "no action" and the review reason was approved. After the review reason was approved, no other action has been taken and the review reason was resolved. The initial decision resolved the review reason after a period. → Good decision
    - The decision was "no action" and the review reason was approved. After the review reason was approved, an action has been taken and the review reason was resolved. The decision to do nothing did not resolve the review reason and an action was required. → Bad decision
    - The decision was "action" and the review reason was approved. After the review reason was approved, no other action has been taken and the review reason was resolved. The initial decision resolved the review reason after a period. → Good decision
    - The decision was "action" and the review reason was approved. After the review reason was approved, an action has been taken and the review reason was resolved. The initial action did not resolve the review reason and other actions were required to resolve the review reason. → Bad decision

The initial decision is the decision whether to take action or not between the review reason trigger date and the resolve date (situation 1) or between the review reason trigger date and the approval date (situation 2). If an exception has not (yet) been resolved, the performance of this exception is unknown.

When applying these rules to this dataset, we observe that between 60% (R24) and 73% (R83) of the decisions that are resolved, are quantified as good decisions. The results are shown in Table 6.

Review Reason	# good decisions	# bad	% good decisions
R24: Projected stock out	62	40	60.8%
R25: Stocked out	53	20	72.6%
R26: Below MOP	62	33	65.2%
R83: Projected inventory below MOP	192	71	73.0%

Table 6: Performance of decisions

2.4 Conclusion

In this chapter the current situation of the SPO department is explained. To do so, first the role of the SPO department and their main software packages are explained. After that the exception handling process is described and a selection of review reasons to focus on is made. For the chosen review reasons, some preliminary analysis is conducted to provide first insights about the current situation.

- SPM contains all processes to keep the right number of spare parts in inventory to meet service levels at minimal costs. The SPO depart in Amsterdam is responsible for the SPM within the EMEA-region for IBM and uses *Servigistics* and *CPPS* for this.

- 34.474 exceptions appeared in one year, divided over 21 review reasons. Four of these review reasons are selected to focus on: R24 (projected stocked out), R25 (stocked out), R26 (below must order point) and R83 (projected below must order point). These review reasons account for 40% of the exceptions.

- It is chosen to look at whether a planner performed a manual adjustment or did nothing. Data analysis revealed that for exceptions of R24, R26 and R83 most decisions are to take action.

- Data is gathered for a period for the exceptions. The data set has a significant number of missing data, which should be dealt with.

- A performance metric is introduced, which qualifies the decisions taken by a planner based on whether the initial decision of the planner resolved the exception. A performance metric revealed that between 60% and 73% of the decisions are qualified as good decisions.

In the next chapter, literature will be examined on data preparation stage and the modelling stage. For the data preparation literature is examined on how to deal with missing values and how to select important features. For the modelling stage, algorithms are evaluated and performance measures of the algorithms are examined.

Chapter 3: Literature review

In this chapter, existing literature relevant for this research is examined. In the first part, the focus is on literature on creating a proper dataset. After that, the focus is on literature on modelling.

3.1 Dataset

In this section, the focus is on literature on creating a proper learn set. Data preparation is an important step of the project. Often fifty to seventy percent of the time and effort spent is used in the data preparation step. In the data preparation step a lot of modifications take place: data to exclude or include is selected, the data is cleaned, new attributes are derived from original data, data is merged, data is reformatted and features are selected. In this section, the focus is on dealing with missing values and feature selection.

3.1.1 Missing values

Missing values are frequently encountered in any kind of datasets and it is often a major problem. To deal with missing data, two approaches could be followed, which are available in SPSS Modeler: deletion, single imputation and multi imputation.

When applying deletion, all cases are deleted for which one or more variable values are missing. Although this is a straightforward and simple approach, Gelman (2006), Baguley (2012) and Kwak and Kim (2017) mention that using a deletion approach may result in an increase of the standard errors due to the smaller sample size. This is especially a problem in a small dataset with a lot of variables. If the missing values differ systematically from the other cases, it could also lead to biased results. Baguley (2012) discourages this approach when there are a lot of cases with missing values. Deletion is the most commonly used method to handle missing values in SPSS Modeler and deletion can be applied by using the *Selection Node*.

Instead of deleting cases with missing values, substituting values can be a solution, which is called imputation. In this way, the complete dataset is kept, which reduces bias. However, other bias is introduced since data is estimated. Imputing missing variables vary from simple approaches to rather complex approaches. Simple approaches may have a lower standard error, since the expected value is pretended to be known (Gelman 2006). He mentions some simple approaches:

- Imputation by logic rules.
- Imputation by using the mean or mid-range.
- Imputation by introducing an indicator for missing values:
    - Categorical: add an extra category which indicates missingness.
    - Continuous: replace the missing value and add a variable which states whether a value is missing.
- Last value carried forward: using a historical value of the same case.
- Using information from related observations.

More complex approaches suppose that variables have a certain predictive distribution and estimates are based on this distribution. Within SPSS Modeler, the only algorithm is C&RT, which stands for Classification and Regression Tree. This algorithm imputes a value when it is missing, based on the distribution of the data for the feature. For statistical machine learning algorithms, often the mean is imputed, such that the imputed value will not have a big influence.

Approach for this research

Since we are dealing with a relatively small dataset, deletion is not preferable. Simple imputation approaches are useful for features for which the expected value is known, which could lead to a lower standard error. For features for which the values of missing data are not known at all, more complex approaches can be used. For this research, deletion approaches will be avoided. Simple approaches and complex approaches will be used for missing data.

3.1.2 Feature selection

Generally, a dataset contains lots of features. Features are measurable properties of the object we are analyzing. Speaking in terms of exceptions, features can be the current inventory level or the vitality of the part for which the exception is triggered, or the number of other review reasons connected to this exception. Selecting relevant features is an important task. According to Liu et al. (2010), feature selection has three main advantages: (1) removing irrelevant and redundant features, (2) reducing the complexity of the data model and increasing the performance of an algorithm and (3) making it more understandable.

To reduce the number of features, different methods are known. Although some authors mention the use of domain knowledge for excluding irrelevant and redundant features (Tan et al., 2006 and Li et al., 2006), different algorithms exist for selecting the best combination of features. Since the number of combinations of features are 2^n, where n is the number of features, which leads to a significant number of combinations, different methods are known in literature to find the most valuable features. Most authors distinguish four methods: *filter*, *wrapper*, *ensemble* and *hybrid* methods.

Filter

Filter methods calculate measures (e.g.: consistency, distance, similarity, statistical measures) to select an optimal feature set. Filter methods take place before the modelling algorithm uses the features. Univariate measures rank the features one by one, in contrast to multivariate measures which ranks subsets of features. Visalakshi and Radha (2014) mentions that filter methods are computationally simple, fast in process and independent of algorithms. On top of that, *filter* methods avoid the problem of overfitting. In a limited time, it produces good and relevant features. The main disadvantage of these methods is the ignorance of interaction effects.

Wrapper

Whereas *filter* methods use performance measures, *wrapper* methods require an algorithm to calculate the performance. These methods measure the performance of different subsets, based on the algorithm's performance. The main advantage is that interaction effects are evaluated as well. However, they are much slower, since the algorithms needs to be performed for every subset. Next to that, results can be biased towards the performance measure of the algorithm (Jovic et al., 2015) and there is a high risk of overfitting (Visalahski and Radha, 2014).

For both *filter* and *wrapper* methods, there are three starting points for subset generation. *Forward selection* (1) starts with an empty set and adds a feature one by one. *Backward reduction* (2) starts with all features and deletes features one by one. *Forward selection* performs best when the optimal subset is small, in contrast to *backward reduction* where the opposite is true. The main disadvantage of (1) is that it is unable to remove features that become obsolete after adding features in a next step. The main disadvantage of (2) is that it is not able to reevaluate features after it has been discarded (Gutierrez-Osuna, n.d.). Bi-directional search (3) uses a combination of (1) and (2), where features selected by *forward selection* are not removed and features removed by *backward selection* are not selected. In this way, it deals with the two main disadvantages of (1) and (2).

Hybrid

Hybrid methods use a combination of *filter* and *wrapper* methods. First, a *filter* method is used to reduce the number of features and after that a *wrapper* method is used to find the best subset. These methods generally have a high performance but are still efficient.

Embedded

In *embedded* methods, the feature selection is embedded in algorithms. In decision trees like C4.5, CART and Random Forest, feature selection is done in the algorithm. In some algorithms (e.g.: LASSO or Elastic Net) feature weighting is performed, which minimizes fitting errors. Penalties are induced to features that do not contribute. A disadvantage of *embedded* methods is that the selected features are only selected in the algorithm and cannot be used

for other algorithms. Next to that, these methods do not always have to perform better. In a research by Pereira et al. (2015), *embedded* methods (LASSO regression and Ridge regression) did not perform significantly different than simple *filter* methods.

Approach for this research

If *decision tree* algorithms are chosen as modelling algorithms, feature selection is not necessarily needed. However, knowing which features are relevant will reduce the complexity of the model and could improve the performance. Since the dataset in this research contains a relatively small number of features, the disadvantage of *wrapper* methods that they are slower is not a problem. Since interaction effects should be taken into account, a *wrapper* method will be used as feature selection method. A bidirectional search is will be used, since it deals with both disadvantages of *forward selection* and *backward reduction*.

3.2 Modelling

In this section, the focus is on how to balance data, different modelling techniques and their performance. In the first part, different approaches on how to balance data are examined. Next, different relevant supervised machine learning techniques are discussed. After that, the focus is on performance measures of algorithms and finally the comparison of different classifiers is discussed.

3.2.1 Algorithms

Different algorithms can be used as modelling technique. However, as the "No Free Lunch" theorem states (Wolpert and Macready, 1997), it is unknown which algorithm will perform best for which problems. Many authors (Kuhn and Johnson (2013), Tan et al. (2006)) suggest using different algorithms to see how the baseline performance is and which algorithm performs well. Kuhn and Johnson (2013) and Kotsiantis (2007) presented frameworks that compare different algorithms and their characteristics. These frameworks can be found in Appendix G and will both be used to find the pros and cons of algorithms.

In the framework by Kuhn and Johnson (2013), five main classification algorithms are distinguished: decision trees, neural networks, k-nearest neighbors, support vector machines and naïve Bayes. Next to that, Kotsiantis (2007) discusses regression algorithms as well.

When comparing these models, the main advantage of using decision tree algorithms is that they handle continuous, binary and discrete data at the same time. Other algorithms often only handle continuous or discrete data and when mixed data is input to the model, the data often needs to be transformed. Next to that, decision trees handle missing values well. Other advantages of decision tree algorithms are the easiness to understand, limited computation time, robustness to noise, embedded feature selection (Gupta et al., 2017). However, in

general the accuracy of other algorithms is better. Most decision trees can be used for both regression and classification purposes.

Neural networks generally perform better than decision tree algorithms. However, computation time is high, the algorithm itself is hard to understand, it is not robust to noise and feature selection and scaling needs to take place before the algorithm starts. On top of that, discrete values need to be transformed into numerical values and neural networks are not robust to feature noise. Neural networks can be used for both regression and classification purposes.

KNN is an easy to understand algorithm and computation time is fast (although the classification itself is slow). Feature selection needs to take place before the algorithm runs and the accuracy is average in general. Discrete values need to be transformed to numerical values before the algorithm can start. Neural networks can be used for both regression and classification purposes.

Support Vector Machines are one of the most popular algorithms and generally perform well. However, the algorithm is hard to explain, and computation time is often high. Although feature selection should be performed before the use of the algorithm, the algorithm itself can cope with redundant and irrelevant features. Support Vector Machines can be used for both regression and classification purposes.

Naïve Bayes algorithms require limited computation time, but the accuracy is limited as well. Although there are ways to deal with mixed data, the algorithm is more suited for categorical and binary data. Continuous data should be binned to categorical data before applying a Naïve Bayes algorithm. Naïve Bayes algorithms do not cope well with redundant and irrelevant data and therefore feature selection should take place before the algorithm runs.

Regression algorithms often requires lots of pre-processing, but the computation time is limited and the algorithms are easy to understand. Feature selection is embedded in some regression algorithms. Disadvantage is that these algorithms are less robust to feature noise and handle continuous input data only. Different regression algorithms exist which can be used for regression, classification or both.

Selection for this research

Since we are dealing with mixed data (data which has categorical as well as numerical data), decision trees are the only modelling techniques which can be used directly for the final dataset, which is a major advantage. Other advantages of using decision trees over other classification techniques are the easiness to understand the algorithm and knowledge gained, the embedded feature selection, the speed of classification and that they require less data preparation compared to other algorithms. Since one of the results of this research is a

working tool, the models build should be easily understandable to implement it into a tool, which makes the former advantage an important one. Due to these reasons, it is chosen to use decision trees as classification algorithms. In Section 5.2 it is found that the C5.0 is the best performing algorithm in a test-design. Therefore, this algorithm will be explained.

C5.0 algorithm

To explain the *C5.0* algorithm, first a decision tree will be explained. A decision tree is a series of questions about the features of a record, until a conclusion can be made about the class of the record.

The *C5.0* algorithm is developed by Ross Quinlan and is a successor of the ID3 and C4.5 algorithm. It is a statistical classifier and can handle both continuous and categorical data, it deals with missing values and it uses pruning to prevent overfitting. The order of questions can be represented in a decision tree, as shown in Figure 14. This decision tree consists of a *root node*, *internal nodes* and *leave nodes*. A *root node* (red) is the starting point of a decision tree and has no incoming arches. *Internal nodes* (green) have one incoming arch and two or more outgoing arches. The *leaf nodes* (blue) do not have outgoing arches and they are assigned to a class. Although decision trees are easy to understand, there are exponentially many decision trees which can be built (Tan et al., 2006). The *C5.0 algorithm* starts with the full set of training data and splits the data consecutively on the attributes which give the highest information gain. The information gain is calculated by subtracting the average information value of the child nodes from the information value of the parent node.

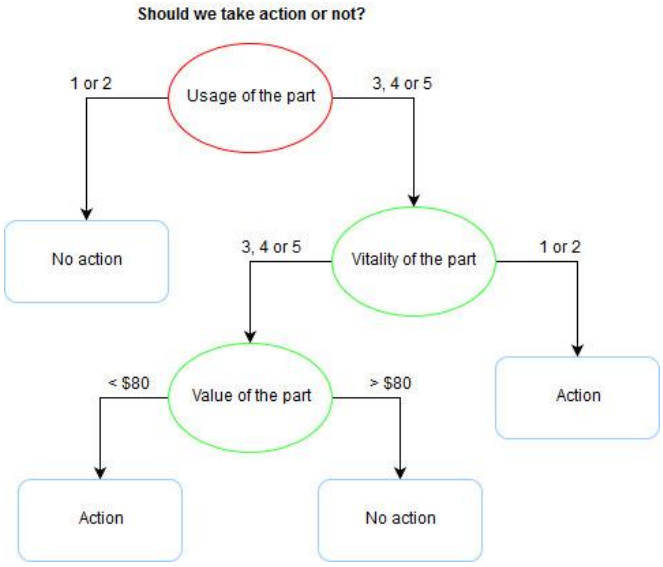

Figure 14: Decision tree example

The *C5.0* algorithm uses *entropy* to calculate the information gain. The *entropy* is calculated by the following formula:

$$entropy = -\sum_{i=1}^{c} p(i|t) * log_2(p(i|t))$$

In this equation, *c* is the number of classes (*action* or *no* action in our case) and *p(i|t)* is the fraction of records belonging to class *i* at node *t*. In Figure 15 the function of the entropy is

plotted. As we can see the entropy value is high if the dataset is equally divided over the classes. The more imbalanced, the lower the value.

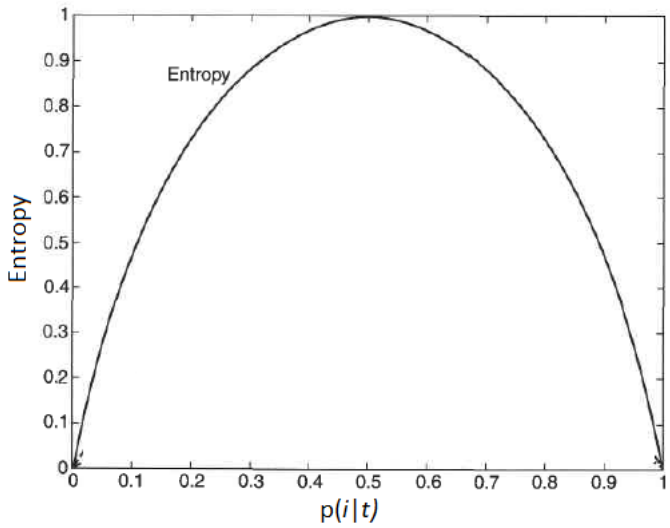

Figure 15: Plot of entropy function (Tan et al. (2006))

To calculate the information gain, first the *entropy* is calculated for the parent node, which is the node where a split is going to be made. All possible splits are evaluated and the *entropy* for all child nodes is calculated. For all possible splits, the *entropy* is calculated for the child node and now the information gain can be calculated by the following formula:

$$Information\ gain = I(parent) - \sum_{j=1}^{k} \frac{N(v_j)}{N} * I(v_j)$$

Where *I()* denotes the entropy, *k* denotes the number of child nodes, *N* is the total number of records in the parent node and $N(v_j)$ is the number of records associated with child node v_j. As can be observed in the formula, the highest information gain is achieved if the distribution is quite evenly balanced in the parent node and the child nodes are quite unbalanced.

The attribute which results in the highest information gain is used as split and the process repeats itself.

The stopping criterion for the *C5.0* algorithm is the minimum number of records in the child node. Aside from this stopping criterion, there are two cases which can occur:

- All records belong to a certain class. In this case, a leaf node is created.
- No feature does give any information gain. In this case, no child node is created and the expected value of the parent node is assigned as class to the records.

After the decision tree is built, pruning can be applied to reduce the decision tree size.

C5.0 in SPSS Modeler

In Figure 16 the *C5.0* algorithm settings available in SPSS Modeler 18.1.1 are shown.

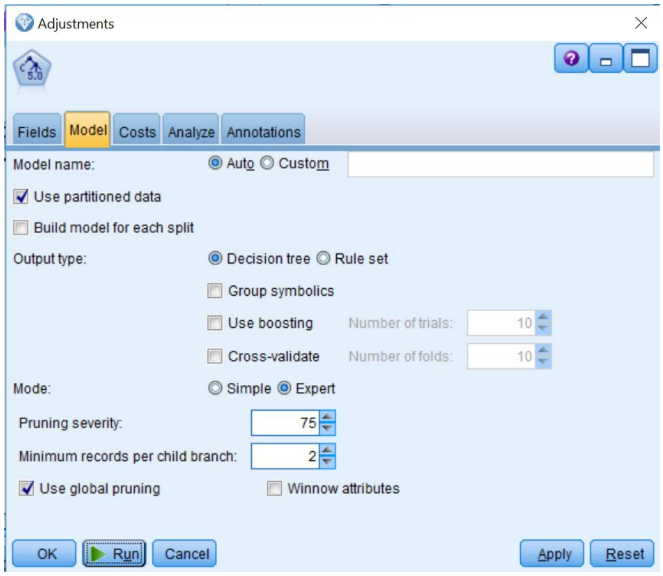

Figure 16: C5.0 settings

The first setting indicates whether *partitioned data* is used or not. Enabling this setting leads to building a model based on the training data and the performance is evaluated on the testing data. If this option is disabled, all data is used for building a model and the performance of the model is evaluated based on the whole dataset.

Using the *build model for each split* setting results in multiple models being built for certain values of features. For example, if we provide one dataset with exceptions of R24, R25, R26 and R83 together, we could make the feature *review reason ID* a split feature, which leads to building models per review reason. These models can be evaluated separately.

The *output type* can be set to a *decision tree* or a *rule set*.

If the *group symbolics* setting is enabled, groups of data that show the same patterns are merged. This is helpful to reduce the complexity of a decision tree.

Boosting can be used to improve the accuracy. This algorithm builds multiple models (*number of trials*). The first model is built in the usual way, but for the second model, the focus is on the misclassified records in the first model. In this way, the models focus on the errors of the previous built model. Finally, voting is used to combine the individual predictions to one final prediction. Using *boosting* for small datasets can result in over-trained models.

Cross-validation is a method which is extremely useful for small datasets (IBM, n.d.-c). This method splits the available data into a number of subsets (*number of folds*) and every subset is used once to evaluate the performance of the models built on the other subsets. The performance is evaluated by taking the averages of the performance indicators of the individual models.

The *mode setting* can be set to *simple* or *expert.* If the *simple* setting is used, there is less control over the control parameters and general settings are applied. One should only indicate

the *% of expected noise* in the data. With the *expert* option, one can define the *pruning severity* and the *minimum number of records per child branch.*

The *pruning severity* is a value between 0 and 100. The higher the number, the more the decision tree is pruned and thus the more concise the tree is.

Next to the *pruning severity*, the *minimum records per child branch* should be set. Setting this value high will lead to a small decision tree. However, setting this value too low, will result in a possible over-trained model.

Using *global pruning* will evaluate all subtrees of the final model to determine weak subtrees. *Global pruning* is performed by default (IBM, n.d.-c) to prevent over-trained models.

The setting *winnow attributes* indicates whether the features are evaluated on relevance before building a model. Enabling this option will decrease the computation time, since only relevant features are evaluated at each node.

3.2.2 Performance measures

To know how well a machine learning algorithm performs, multiple performance measures are known in literature. A simple example is given to clarify the performance measures. Therefore, a confusion matrix is shown in Table 7.

Since some review reasons are dealing with the decision whether to take action or not, we clarify the use of the confusion matrix based on this example. A machine learning algorithm is fed with input data and the decision is to take action or not. Since a labeled dataset is fed to the model, we already know what the decision should be, but the algorithm will try to learn from a part of the data and uses the knowledge gained to predict the outcome for the other part of the data. If, for example, the algorithm classifies a certain input with "No action" and the actual decision was "action", the model is obviously wrong. After a dataset is run, the confusion matrix summarizes the number of instances predicted (in)correctly.

		Predicted class	
		Action	No action
Actual class	Action	TP	FN
	No action	FP	TN

Table 7: Confusion matrix

The following terminology is used for the confusion matrix:

- **TP** (true positive): the algorithms is correctly telling you to take action.
- **TN** (true negative): the algorithm is correctly telling you not to take action.
- **FP** (false positive): the algorithm is falsely telling you to take action.
- **FN** (false negative): the algorithm is falsely telling you not to take action.

Based on the confusion matrix, the following performance measures can be calculated, which are explained subsequently: *accuracy, precision*, *recall, F_1-score* and the *Area Under the Curve (AUC*).

Accuracy

One of the most general performance measures to use is the *accuracy*. The *accuracy* is measured by dividing the correctly classified instances by the total number of classified instances:

$$accuracy = \ (TP + TN)/(TP + TN + FP + FN)$$

A major drawback of this performance measure is that it treats all classes as equally important (Brownlee (2015), Sunasra (2017), Tan et al. (2006)). This is not a problem when the dataset contains an equal number of instances per class, but it can be problematic when the dataset is imbalanced. Take for example a company with a production line which performs at six sigma level[1]. A machine learning algorithm will be able to have an *accuracy* of 0.9999 of the cases if it will just classify every instance as "no defect". This is called the *accuracy paradox*. In these cases, correctly predicting the "defect" has much more value than predicting the "no defect" classes. The *accuracy* performance measure does not deal with this problem. To deal with this problem, other performance measures are available. Another option is to balance the data in the data preparation phase, which will be explained in 3.2.

Precision, recall and F_1-measure

For the following performance measures, the performance should always be measured for the minor class (the class with the least observations). In the following example this means that the "action" class is supposed to be the minority class.

The *precision* is measured by dividing the number of positive predicted instances which are actually positive (TP) by the total number of positive predicted instances (TP + FP). In our example this is the number of instances predicted as "action" which are actually "action" divided by the total amount of instances predicted as "action".

$$Precision = \ TP/(TP + FP)$$

The *precision* score is a real number between 0 and 1 and it holds that the higher the *precision*, the less false positive errors are made.

Recall on the other hand measures the fraction of positive examples correctly predicted as positive. In the example this is the number of instances predicted as "action" which are actually "action" (TP) divided by the total number of actual "action" (TP + FN).

[1] Six sigma level means producing at 3.4 defects per million products.

$$Recall = \ TP/(TP + FN)$$

The *recall* score is a real number between 0 and 1 and it holds that the higher the *recall*, the less misclassified positive examples.

It is possible for algorithms to maximize one of these performance measures. To prevent this from happening, another performance measure is being used which deals with both *precision* and *recall*: the F_1-score. This performance measure is a harmonic mean between *precision* and *recall*, which tends to be closer to the smaller one.

$$F_1 = \frac{2}{\frac{1}{recall} + \frac{1}{precision}}$$

Since the value of *recall* and *precision* both lie between 0 and 1, the value of F_1 lies between 0 and 1 as well. The higher the number, the better both *recall* and *precision* scores are.

ROC curve and area under the curve (AUC)

The final performance measure to discuss is the *area under the curve (AUC)*. This performance measure gives the capability of a model of distinguishing between classes, the separability. The higher the AUC, the better the separability of the model. The *AUC* will be explained by an example.

For most tests, a perfect separation of classes will not be observed and overlap between the classes will occur, which can be seen in Figure 17. In this figure, the green distribution represents the *action* class and the red distribution the *no action* class. If we set the *criterion value* as depicted in Figure 17, we predict all values to the right of the *criterion value* as *action* (positive) and all values to the left of the *criterion value* as *no action* (negative). This will result in the four classes: *TN, FN, FP* and *TP*. For all *criterion values*, the *sensitivity or precision* (the fraction of positive examples correctly predicted as positive) and the *specificity* (the fraction of negative examples correctly predicted as negative) are calculated. *A* lower *criterion value* will lead to less *FN* and thus a higher *sensitivity*, but it will increase the number of *FP* and thus will lead to a lower *specificity*.

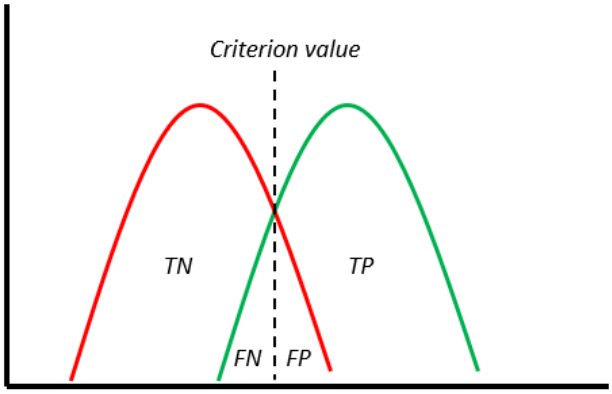

Figure 17: Example distribution of classes

If we vary the *criterion value,* we obtain different values for the *sensitivity* and the *specificity*, as can be derived from Figure 17. To calculate the AUC, the ROC-curve should be plotted. The ROC curve plots the *sensitivity* on the y-axis and (1 – *specificity)* on the x-axis, for different *criterion values*. In Figure 18 an example of a ROC-curve can be seen. The blue line gives the performance of the model. A model which makes random guesses resides along the red line. The closer to the upper left corner, the better the model can distinguish different classes and thus the performance of the classifier. The *AUC-value* is equal to the surface below the blue line.

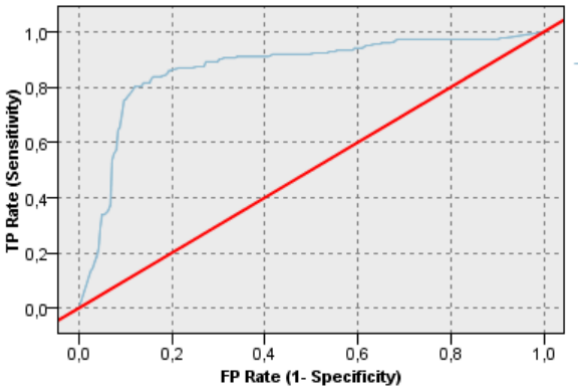

Figure 18: Example ROC-curve

3.3 Conclusion

After examining current literature, we know how to deal with missing values, we know how to select features to focus on, we know which classification algorithms are known, we know how the performance of models made by algorithms can be measured. The following conclusions are found:

- Missing features should be imputed and deletion should be avoided.
- A *wrapper* method will be used to select only the relevant and important features. This will decrease the complexity of the model.
- Decision trees algorithms will be used for this research, since they can deal with both categorical and continuous data, building models goes relatively fast and results are easy to interpret.
- All performance metrics are valuable, since they all measure other performance criteria. For balanced datasets the *accuracy* is a good performance measure. The other performance metrics become more valuable if the dataset is imbalanced.

Chapter 4: Data preparation

In the previous chapters, a selection of review reasons was motivated, an initial dataset is gathered, performance metrics were derived for actions taken by the planners, and the literature has been reviewed. In this chapter, we first integrate the data to create one dataset. After that, the data is cleaned by imputing missing data, and new features are constructed from the initial data. Next, the decision whether to train the model to replicate the decision of the planners (train based on all observed exceptions) or to make good decisions based on the performance metrics defined in Section 2.3.2. Also, the decision whether the cleaning step is performed or the decision tree algorithm should deal with missing values itself will be addressed. Finally, valuable features are identified with a feature selection approach. As a result, we focus on the third step of the CRISP-DM methodology in this chapter, as shown in Figure 19. The results of this step will enable us to continue with the modeling stage in the next chapter.

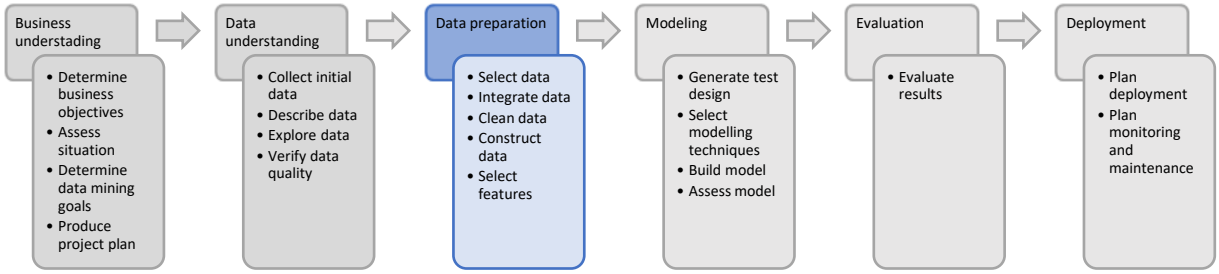

Figure 19: Current focus within CRISP-DM methodology (Chapman et al, 2000)

4.1 Integrating data

Since the data is extracted from different databases, SPSS Modeler is used to merge the data to one data file. Per exception, the corresponding data of features is merged from different tables. As defined in Section 2.3.1. the features can be categorized into *part characteristics, inventory information, order information, settings* and *exception related.*

The output is a file containing an overview of all exceptions with values for all features. The features and their explanation can be found in Appendix C and the model that merges all data into one file can be found in Appendix E.

4.2 Cleaning data

As explored in section 2.3, the dataset contains missing values and one of the activities in step 3 of the CRISP-DM methodology is imputing missing data. Hence, the features that have missing values are evaluated in this section. The results are shown in Table 8. Note however, that in Section 5.1 we perform experiments with both, the imputed dataset and the original dataset. Thus, we will further evaluate whether the imputation step is recommendable for IBM.

Feature	% complete	Imputation strategy
Shelf life (years)	2.3%	For the feature *shelflife* only parts which have a shelf life will have a value. In the dataset we observe a significant number of missing values for the *shelf life*. We may impute the missing values by applying a logic rule that assigns a high value (for example 20 years) to missing values. However, as will be discussed in the next section, we chose instead to change this feature from a continuous feature to a categorical feature.
Open order arrival	37.4%	For the feature *open order arrival* only parts which have an open order will have a value. We impute the missing values by applying a logic rule that assigns a high value (in this case two times the maximum value found in other records) to missing values, which indicator that it will take a very long time before a next order arrives. To isolate unwanted effects of imputing values, an imputation indicator is added, which will be explained in the next section.
Forecast method	65.6%	For the forecast method, two cases exist: (1) there is no forecast and therefore the forecast method contains a null value or (2) there is a forecast, however a forecast method is missing. The first case is solved by adding the category *No forecast* to the forecast method. For the second case, the category *Undefined forecast* is added.
Lead time (weeks)	72.3%	A missing value for this feature implies that no supplier is available. Since no supplier is available, we choose to use a logic rule: a missing value is imputed by a high value, in this case two times the maximum value found for other cases.
Return rate	86.2%	A missing value for the return rate means that no demand or returns have been recorded in the last two years. This value is imputed by the mean of all observed values for the *return rate*. Next to that, the binary (yes/no) feature *return rate indicator* is added which states whether the feature *return rate* had initial values. This is done to isolate unwanted effects of imputing a missing value by the mean.
PSS	87.0%	A missing value for *PSS* means that for this part currently no predictions of demand are made and therefore no tactical inventory levels exist.

		Therefore, the mean of the *PSS of the parts of* other exceptions is imputed and a new binary feature is added which states whether the part was planned (yes/no), again to isolate unwanted effects of imputation.
First stock date	89.5%	A missing value for *stock date* implies that the part is not yet stocked. Therefore, the value of today will be imputed (2019-01-01). However, as will be discussed in the next section, this feature will be transformed to the number of years difference between the year of the exception and the *first stock date*.
Usage x price	90.2%	This feature categorizes the value of the part by making a tradeoff between the usage of a part and the price of a part. Since for most parts we know the historical usage and the *weighted average cost* value, we multiply these values and use the known *Usage x price* values in an algorithm to predict missing values for this feature. The C5.0 algorithm in SPSS Modeler is used, which predicts 85% of this feature right. For the cases where the algorithm is wrong, it predicts the value between one and two values away from the actual value, with a total average deviation of 0.21 values away.
Usage	90.2%	See *Usage x price* above. The algorithm uses *historical activity 2 years* and *usage* of other records to derive the values. The C5.0 algorithm in SPSS Modeler is used, which predicts 76% of this feature right. For the cases where the algorithm is wrong, it predicts the value about one value away from the actual value, with a total average deviation of 0.10 values away.
EOS date	91.0%	For all parts which have a *birth date,* often an *EOS date* is calculated. If a missing value is encountered, it is replaced by the maximum value found for other exceptions. Next to that, an indicator will be added which indicates whether the feature had an initial value for this part. However, as will be discussed in the next section, this feature will be transformed to number of years difference between the year of the exception and the *EOS date*.
Division group name	98.8%	A missing value for *division group name* is imputed by a new category *No value*.

Birth date	99.5%	Since a missing value of the feature *birth date* implies that the part does not exist, records without a birth date are removed from the dataset.
Division owner code	99.6%	See *Division group name*.
A-code	99.6%	This feature specifies where this part should be stocked. If this is unknown, the part does get a *no recommendation* value. However, we only encountered a few missing values.
C-code	99.6%	This feature specifies how a part should be ordered. A value of *0* means that the order method is undefined. Therefore, a value of *0* is imputed for missing values.

Table 8: Imputations

After imputing missing values according to the given strategy in Table 8, our dataset contains values for all records and features.

4.3 Feature construction

In this section, new features are created. Based on information which is not yet covered by the current features, new features are created. These features are created by using multiple features or by changing the data type of features. The new features, their measurement type and why and how they are created are explained in this section. A summary is provided in Table 9.

Birth age, first stock age and EOS age

These features are currently continuous features expressed in a date. To distinguish recent parts and old parts, we choose to change the *birth date* to a value which indicates how many years old the part is. This is done by computing the difference in years between the date of the exception and the *birth date*. It is chosen to perform the same approach for the *first stock date* and the *EOS date*. This approach results in the new continuous features *birth age, first stock age* and *EOS age*.

Life cycle point

Currently, no feature is available which provides information in which stage of the life cycle a product is. Since it can be assumed that other decisions will be taken for a part which is in the beginning of its life cycle compared to a part which is at the end of its life cycle, we introduce the *life cycle point*. The *life cycle point* is calculated by dividing the *birth age* by the difference in year between the *EOS age* and the *birth age.* This results in the continuous feature *life cycle point*. Values for this feature are between 0 and 1.

Shelf life-, open order- and supplier indicator

Since only a few parts have a shelf life, the value of the *shelf life* is less relevant; therefore, it is chosen to create the feature *shelf life indicator* which indicates whether a part has a *shelf life* or not. Next to that, we introduce the binary feature *open replenishment order* that indicates whether there is an outstanding order. Finally, currently there is no feature that specifies whether a supplier is available. Since a missing supplier may indicate a problem, we decide to introduce the binary feature *supplier indicator*.

Inventory position

Although the inventory level is currently used in the dataset, the inventory position does give additional information. The difference between the *inventory position* and *inventory level* is whether open replenishment orders are included or not. If (a substantial number of) parts are on order, it could be the case that a different action is taken. The inventory level alone could lead to a misrepresented situation. Therefore, we add the *inventory position* (*inventory level* + replenishment open order quantity) as a feature.

Fraction inventory position currently available

In addition to the *inventory position*, it is valuable to know what fraction of the *inventory position* is currently available. If most parts are not available yet, this perhaps would result in different decisions than for a situation where the inventory position is equal to the inventory level. This fraction is calculated by dividing the inventory level by the inventory position. If backorders exist, this fraction is set to 0. Although the *inventory level* gives information about the on-hand stock, this feature gives information about the number of parts on open order compared to the parts on-hand. On top of that, this feature is comparable between parts.

Fraction recent demand

Currently, there is no feature that indicates how the recently observed demand behaves compared to the demand over a longer period. If the recent demand is relatively high, perhaps one may choose a different action if compared to a situation for which the recent demand is relatively normal. Therefore, the feature *fraction recent demand* is introduced, which divides the historical demand during the last four weeks by the historical demand during the last two years.

Critical severity

To give more information about the quantity that is or will be critical, the feature *critical severity* is introduced. The *critical severity* divides the critical quantity by the *PSS.* The critical quantity is the number of pieces deemed to be critical in the supply versus the need of the part. Since tactical levels are part specific, the *critical severity* is a good feature which makes it possible to compare the critical problem between different parts.

Runout time

The stock level does not provide much information on its own. Therefore, the number of weeks is calculated for which supply is available, which gives an indication when the problem will occur. This is done by dividing the *inventory level* by the *forecasted quantity*. Since the *forecasted quantity* is the forecast for two years and we want to calculate the number of weeks supply left, the ratio *inventory level / forecasted quantity* should be multiplied by 104 (number of weeks in 2 years).

Inventory position severity

Until now, there is no feature that relates the inventory position to the predefined *PSS*. However, such a feature may help to identify the severity of the situation. If the current inventory level is low and no parts are on order, this feature has a low value, which indicates the severity of the problem. Therefore, we introduce the feature *inventory position severity*. The *inventory position severity* is a ratio which divides the *inventory position* by the *PSS*. In this way, it is known whether the *inventory position* is lower (< 1) or higher (> 1) than the *PSS* and it makes it possible to compare different parts.

We summarize the new feature in Table 9.

Feature	Measurement	How is it calculated?
Birth age	Continuous	The *birth age* is calculated by taking the difference in years between the date of the exception and the *birth date*.
First stock age	Continuous	The *first stock age* is calculated by taking the difference in years between the date of the exception and the *first stock date*.
EOS age	Continuous	The *EOS age* is calculated by taking the difference in years between the *EOS date* and the date of the exception.
Life cycle position	Continuous	The *life cycle point* is calculated by dividing the *birth age* by the difference in year between the *EOS age* and the *birth age.*
Shelf life indicator	Binary	The *shelf life indicator* is a binary value the value is *yes* when a part has a *shelf life* and *no* when it does not.
Open order indicator	Binary	The *open order indicator* is a binary value the value is *yes* when a part has *open replenishment orders* and *no* when it does not.
Supplier indicator	Binary	The *supplier indicator* is a binary value the value is *yes* when at least one supplier type is available for a part and *no* otherwise.

Inventory position	Continuous	The *inventory position* is calculated by adding the *open order quantity* to the *inventory level (stock).*
Fraction inventory position currently available	Continuous	The *fraction inventory position currently available* is calculated as follows: (*inventory position – open order quantity) / inventory position.*
Fraction recent demand	Continuous	The feature *fraction recent demand* is calculated by the historical demand during the last four weeks by the historical demand during the last two years.
Critical severity	Continuous	The feature *critical severity* divides the *critical quantity* by the *PSS*.
Runout time	Continuous	The feature *runout time* is calculated by dividing the *inventory level* by the *forecasted quantity* and multiplying it by 104 (to get the number of weeks left).
Inventory position severity	Continuous	The feature *inventory position severity* is a ratio which divides the *inventory position* by the *PSS*.

Table 9: New features

After the data cleaning step and the feature creation step, the dataset consists of 3.254 records with 61 features, which are shown in Appendix H . This dataset will be split into four datasets, for each review reason one, which is shown in Table 10. Per review reason, the number of records is given after the data cleaning step. Next to that, the percentage of exceptions on which a good decision has been taken, according to our performance indicator explained in 2.3.2, is given.

Review reason	# records	% good decisions
R24	648	74.1%
R25	495	80.3%
R26	460	73.8%
R83	1.651	77.2%

Table 10: Datasets overview

4.4 Select data

In the exception gathering period of one-and-a-half month, about 3300 exceptions were observed. Using all exceptions in a machine learning algorithm will lead to a model which tries to reproduce the decisions from the planners, which would make the process more efficient. However, since the goal of the assignment is to make it more effective as well, only exceptions for which it was evaluated (cf. Section 2.3.2) that a good decision has been taken, should be considered. Hence, we can distinguish between two data sets: one that contains all exceptions and one that only contains exceptions with good decisions. The former dataset may be used

to improve the efficiency and the latter dataset may be used for both, the efficiency and the effectiveness. We summarize this situation in Figure 20. A possible downside of using only exceptions on which a good decision has been taken, is that specific exceptions are not included in the dataset, since these exceptions are never classified as good decisions.

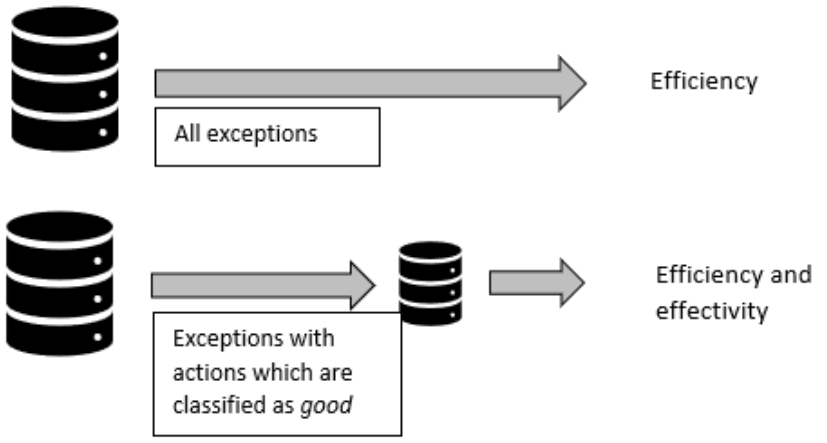

Figure 20: Selecting data

As mentioned in Section 4.2, decision tree algorithms can handle missing data well. However, since we may have more knowledge about the missing values, we chose to impute missing data to get one final and complete dataset. To see in what way this has an influence, we will experiment with datasets where missing data is imputed as motivated in Section 4.2 and datasets where missing data is not imputed. In this way we get four different datasets per review reason, as shown in Table 11, for which experiments are performed in Chapter 5.

Which decisions? ***Imputation yes/no?***		**All** **Imputation**	**All** **No imputation**	**Good** **Imputation**	**Good** **No imputation**
R24	# records	648	650	341	342
	Action yes / no	331 / 317	332 / 318	154 / 187	155 / 187
R25	# records	495	508	233	236
	Action yes / no	187 / 308	191 / 317	92 / 141	93 / 143
R26	# records	460	463	293	295
	Action yes / no	222 / 238	223 / 240	112 / 181	113 / 182
R83	# records	1652	1658	995	998
	Action yes / no	802 / 850	807 / 851	419 / 576	422 / 576

Table 11: Datasets statistics

4.5 Feature selection

To identify the most valuable features and to decrease the complexity of the model by removing irrelevant features, the *feature selection* tool of WEKA is used to analyze the datasets. WEKA is a collection of machine learning algorithms for data mining and is developed by the University of Waikato in New Zealand. Although there is an overlap with SPSS Modeler in functionality, WEKA possesses more comprehensive feature selection methods.

However, based on our research we conclude that using feature selection before the modelling step, has barely an effect on the performance of the models built. Therefore, we leave the approach out and discuss the results right away. The approach to identify the most valuable features can be found in Appendix I.

Based on this approach we found that a significant number of features is not relevant. This approach is performed per review reason and the number of irrelevant features per review reason and dataset are shown in Table 12.

	All Imputation	**Only good Imputation**	**All No imputation**	**Only good No imputation**
R24	19	31	22	29
R25	31	40	27	38
R26	30	25	27	26
R83	23	28	21	28

Table 12: Number of irrelevant features

Overall, we found the most relevant (left part of the table) and irrelevant features (right part of the table), which are shown in Table 13.

Feature	# selected
AnalyzerCode	3.8125
WACValue	3.0625
ReturnRate	2.9375
LifeCyclePosition	2.8125
ForecastedQuantity	2.75
FractionRecentDemand	2.6875
FirstStockAge	2.625
BirthYearAge	2.5625
InventoryPositionSeverity	2.5625
HistoricalActivity2Years	2.5
StockLevel	2.375
Weekend	2.25
HistoricalActivity1Month	2.125
PSS	2.0625
InventoryPosition	2

Feature	# selected
AirSupportIndicator	0
BirthDate	0
FirstStockDate	0
GSupplier	0
StockedIndicator	0
ACode	0.0625
CCode	0.0625
PlannedIndicator	0.125
BSupplier	0.1875
CriticalPartUnsatisfiedIndicator	0.1875
DivisionOwnerCode	0.1875
ForecastMethod	0.1875
NBOIndicator	0.25
Shelflife	0.3125
Successor	0.3125

Table 13: Most relevant and irrelevant features

We observe that the least relevant features can be categorized as part characteristics. For some of these characteristics it could be expected that it has no or only little impact on the decision, such as the *AirSupportIndicator*. For the features *CCode, ACode, BirthDate* and *FirstStockDate*, other features exist which contain similar information and therefore it was no surprise these came out as irrelevant features.

On the contrary, we find that the most relevant features contain features which give information about the supply chain status (e.g. *PSS, InventoryPosition, InventoryPositionSeverity* and *StockLevel*). Some interesting findings are the relevance of the features *AnalyzerCode* and *Weekend*. The *AnalyzerCode* is a code which contains information about the planner, the part brand and the part category and came out to be the most important feature. It could be the case that different planners make different decisions, but since the *AnalyzerCode* does contain more information than the planner itself, we cannot conclude this. Another explanation that the *AnalyzerCode* came out as one of the most important features, is that there are more than 50 unique values for this feature. Since the size of the dataset is limited, it could be that the values indicate whether an action should be taken or not. The *Weekend* feature is introduced since some data in *Servigistics* is only updated in the weekends and based on recalculations, exceptions are triggered. It could be that whether an action is taken or not, depends on whether the exception was triggered by a recalculation or not.

The full results can be found in Appendix I as well.

4.6 Conclusion

In this chapter, the data preparation step of the CRISP-DM methodology is performed, which led to the final datasets. These datasets can be used for modelling. The following tasks were performed this chapter:

- We use deletion, simple imputation techniques and complex imputation techniques to impute missing data for 15 features.
- We construct 13 new features based on existing features.
- We create four datasets per review reason. That is, one dataset version allows missing values while another uses imputed data. Also, one dataset version contains all exceptions while another uses only exceptions for which actions were identified as good decisions.
- Using feature selection algorithms, we determined the relevant set of features for each dataset.

Chapter 5: Modelling

After the final datasets are constructed in the previous chapter, we select the machine learning algorithm in this section. Therefore, we first discuss the experimental setup that is used to identify the most suitable machine learning algorithm for our problem setting. Next to that, the candidate modelling algorithms are selected and the modelling assessing measures are given. The fourth step of the CRISP-DM is the focus of this chapter, as can be seen in Figure 21.

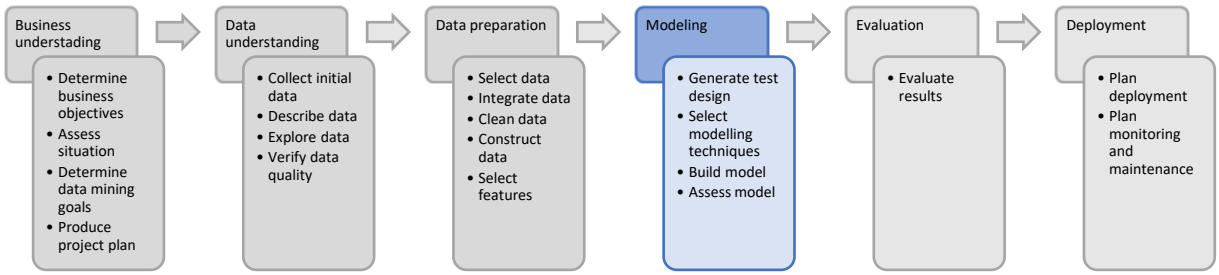

Figure 21: Current focus within CRISP-DM methodology (Chapman et al., 2000)

5.1 Experimental design

The modelling step will be performed on the datasets presented in section 4.4, which are summarized in Table 11. Per dataset the algorithms are performed twice: once on the dataset on which feature selection is performed and once on the initial dataset without feature selection. This leads to 8 experiments per review reason, which are summarized in Table 14.

Experiment	**All exceptions / Good decisions**	**Impute missing data?**	**Feature selection?**
1	All	Yes	Yes
2	All	Yes	No
3	All	No	Yes
4	All	No	No
5	Good	Yes	Yes
6	Good	Yes	No
7	Good	No	Yes
8	Good	No	No

Table 14: Experiments overview

5.2 Candidate modelling techniques

In section 3.3 it is chosen to use decision tree algorithms as classification technique. In SPSS Modeler the following decision tree algorithms are available:

- C5.0
- C&R Tree
- CHAID
- QUEST
- Random Trees

- Tree-AS
- XGBoost Tree

As the "No Free Lunch" theorem states (Wolpert and Macready, 1997), it is unknown beforehand which algorithm will work best for which problem. Hence, different algorithms should be used to see which one performs well for your problem. We used the *auto classifier* in SPSS Modeler, which evaluates the selected algorithms under different settings. The *auto classifier* only gives the *accuracy* per model and since all datasets are quite balanced (the classes *action/no action* are equally present in the datasets), the *accuracy* (cf. Section3.2.2) will give a good representation of the performance and is thus used as performance measure to select the best algorithm.

Data is partitioned into a *training set* and a *test set*. An algorithm uses the *training* dataset to create a model and the *test* dataset to measure the performance. The percentage data assigned to the two sets should be well considered. The more data is used as *training data*, the better a model is able to learn from the data and the model has a better representation of the real world. However, the *test set* will be smaller, which leads to less confidence in the results, since the performance is measured on less records. A common and often well performing practice is the 80-20 practice (Srinidhi, 2018 and A., 2012). In this practice, 80% of the data is used for training and 20% of the data is used for testing.

The *auto classifier* gives the results shown Appendix J. Based on these results the best performing algorithm is selected. The average *accuracy* is calculated, the number of times an algorithm is present in the top 3 is determined and the number of times the algorithm is in the first, second and third position is given. This is shown in Table 15. The *auto classifier* of SPSS Modeler does not provide comprehensive results, however, in Chapter 6 a more in-depth analysis will be given on the results.

Algorithm	Accuracy	# in top 3	*# on 1st*	*# on 2nd*	*# on 3rd*
C5.0	65.3%	18	*8*	*8*	*2*
Tree-AS	64.2%	18	*7*	*7*	*4*
XGBoost	63.9%	20	*6*	*5*	*9*
C&R	63.9%	10	*2*	*3*	*5*
Random trees	63.2%	13	*7*	*2*	*4*
C&R	63.9%	10	*2*	*3*	*5*
QUEST	63.2%	8	*0*	*1*	*7*
CHAID	62.5%	9	*2*	*6*	*1*

Table 15: Number of times algorithm is in top 3

Based on these results it is chosen to use the C5.0 algorithm as modelling technique. Although the *XGBoost* algorithm is most often present in the top three performing algorithms, the *C5.0* algorithm is also often represented in this top three. On top of that, the *C5.0* algorithm is more

often on the first and second place and the average *accuracy* is higher. The *Tree-As* algorithm has a lower *accuracy* and is less often on the first and second place. *The* C5.0 algorithm has been explained in section 3.2.1.

5.3 Parameter settings C5.0 algorithm

Now the algorithm being used is known, the model building settings are chosen. In Section 3.2.1. the settings are explained. In this section, the values of the settings are motivated.

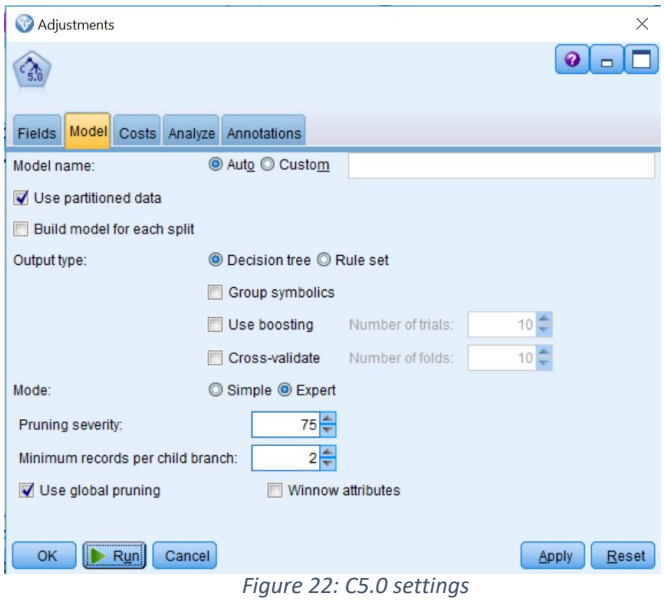

Figure 22: C5.0 settings

Since we are dealing with small datasets, we use *cross-validation* to get more accurate performance measures of the models. However, since the *cross-validation* option in SPSS Modeler does limit us in the evaluation of the models, we built it ourselves in SPSS Modeler. Therefore, *cross validation* is disabled and *partitioned data* is enabled, since we are dealing with a *training*- and a *testing* dataset. In this research it is chosen to use *5-fold cross validation*. The dataset is split into 5 datasets, so called *folds*. Of these *folds,* 4 are used to build a model and the fold left is used as validation set. In this way, 5 models are built where every fold is used as validation set only once. We have chosen for *5-fold cross validation,* since the ratio of the *training* and *validation* dataset is in this situation 80/20. The use of cross validation is explained in section 5.4.

The option *build model for each split* is disabled, since we already split the data per review reason and we further do not distinguish different datasets.

We want to build a *decision tree* rather than a *rule set*, since it is easier to interpret.

The *group symbolics* setting is disabled. If groups of data show the same patterns, these groups will not be merged in this way and the decision tree keeps its distinctive power.

Boosting is disabled since we are dealing with small datasets and we want to prevent over-trained models.

The *mode setting* is set to *expert* option, since we will have more control over the training parameters.

The *pruning severity* is a value between 0 and 100. The higher the number, the more the decision tree is pruned and thus the more concise the tree is. To see what influence the *pruning severity* has on the models, we have chosen to perform experiments with two settings: a low value of 25 and a high value of 75. A value of 75 is the default settings of SPSS Modeler. Since this value is quite high, we added a lower value of 25.

Next to the influence of the *pruning severity*, the influence of the *minimum records per child branch* is evaluated, by using two settings: a low value of 2 and a somewhat higher value of 5. In combination with the *pruning severity,* this creates 4 experiments.

The setting *winnow attributes* is disabled, sine we are dealing with a small dataset and computation time is not a problem.

The settings are summarized in Figure 23.

Parameter	Value
Use partitioned data	Enabled
Output type	Decision tree
Group symbolics	Disabled
Use boosting	Disabled
Cross-validate	Disabled in settings, enabled by modelling myself
Mode	Expert
Pruning severity	25 / 75
Minimum records per child branch	2 / 5
Use global pruning	Enabled
Winnow attributes	Disabled

Figure 23: C5.0 modelling settings

5.4 Assess model

Since we are working with a small dataset, the data which is used as training data will have a high impact on the decision tree built. Therefore, we used cross validation to build more than one model and the performance is measured by taking the average of the models. Next to that, we combined the 5 models built by *cross validation* to build an ensembled model. Since the ensembled model uses multiple models, this model will probably be less sensitive to one particular model built on outliers, which is a possible disadvantage due to the limited size of the data sets. In order to perform *cross validation* and to be able to measure the performance of the ensembled model, we split the original data into 6 datasets: First the data is split into a *training* dataset (80% of the data) and a *test* dataset (20% of the data). The *test* dataset is used for measuring the performance of the ensembled model. The records in the *test* dataset have never been seen by the models. The *training* dataset is used for *5-fold cross validation*, which is explained in the previous section. Therefore, the *training* dataset is split into 5 datasets, so

called folds. A model is built on 4 of these folds and the remaining fold is used for validation. Each fold is only used once as *validation set* and therefore 5 models are built. The splitting of data can be seen in Figure 24.

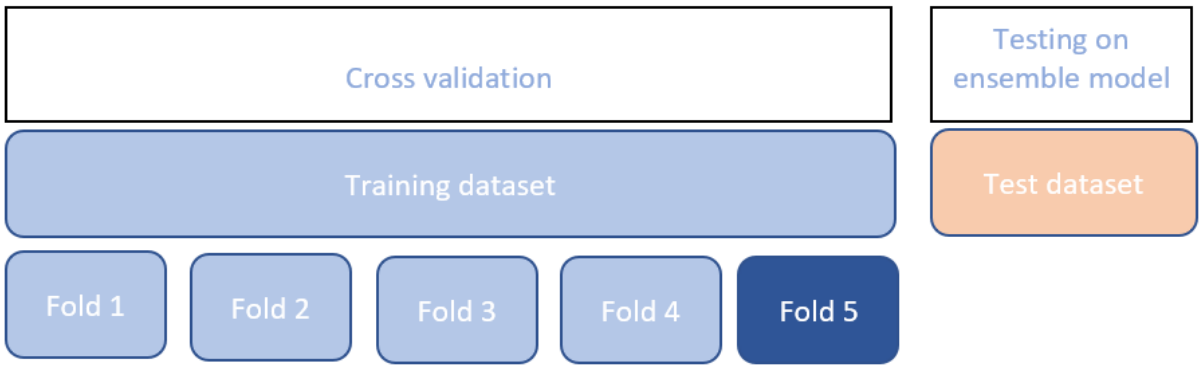

Figure 24: Splitting dataset

To measure the ability of a model to predict whether action should be taken or not, the following performance measures are used: *accuracy, AUC* and *precision*. Individually, these performance measure do not give representative results, since they can be biased. However, using these together, give a clear view on the performance of a model.

First of all, since the datasets are quite balanced, as showed in Table 11, and therefore the *accuracy paradox* discussed in section 3.2.2 is not a problem, the *accuracy* is a proper performance measure for the overall performance of a model. As a reminder, the *accuracy* measures the percentage of correct classified records by the model and is calculated by the following formula:

$$accuracy = \ (TP + TN)/(TP + TN + FP + FN)$$

Next to the *accuracy*, the *AUC* is used as a performance indicator as well. The *AUC* gives valuable information by providing the separability of classes and is equal to the surface below the *ROC-curve* (cf. Section 3.2.2). If the *AUC* is about 0.5, it indicates that the model is not able to distinguish between classes and actually puts all the records into one single class. The *accuracy* and *AUC* together give valuable information about the performance of the overall model, by providing the ability of the model of correctly classifying records and the ability of distinguishing classes.

Finally, the *precision* of the *no action* class is used as performance measure as well, since IBM is interested in reducing the number of review reasons by excluding exceptions on which no action should be taken. Therefore, next to the *accuracy* and the *AUC*, the *precision* is used as performance measure, which gives the fraction of correctly predicted *no action* exceptions out of all predicted *no action* exceptions. This measure is calculated by the following formula:

$$precision = \ TP/(TP + FP)$$

5.5 Conclusion

In this chapter, the modelling step of the CRISP-DM methodology is performed. Based on the experiments and the chosen modelling techniques, the results are evaluated in the next chapter. The following conclusions are found in this chapter:

- Per review reason 8 experiments will be performed. This leads to a total of 32 experiments.

- The *auto classifier* of SPSS Modeler is used to determine which of the 7 decision tree algorithms perform well. Based on these results the C5.0 algorithm is used as modelling algorithm and the different settings for this algorithm are motivated. *5-folds cross validation* is used on which the performance is evaluated by taking the averages of the 5 models. Additionally, these 5 models are combined to one ensemble model on which the performance of never seen data is evaluated.

- The *accuracy, precision* and *AUC* are identified as most suitable to measure the performance in the next chapter.

Chapter 6: Evaluation

After the modelling algorithms are chosen, settings are set, and the performance measures are determined, in this chapter the results are evaluated. The fifth step of the CRISP-DM is the focus of this chapter, as can be seen in Figure 21.

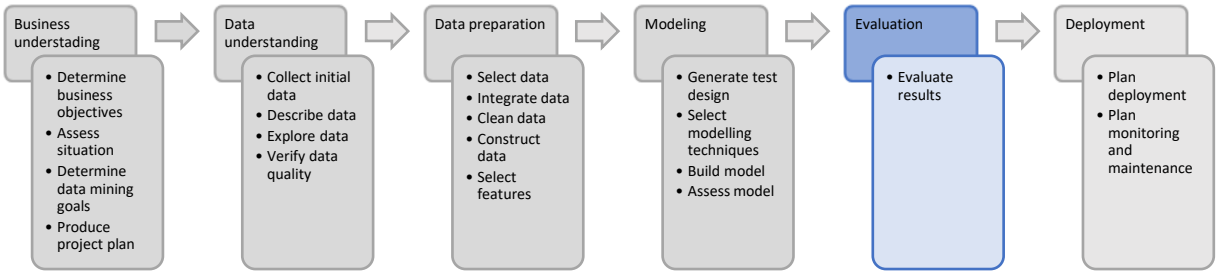

Figure 25: Current focus within CRISP-DM methodology (Chapman et al., 2000)

6.1 Evaluate results

In this section, the C5.0 algorithm is run with 4 settings (cf. Section 5.3) on the 8 experiments (cf. Section 5.1) per review reason and the results are evaluated. The evaluation of the results is divided into 3 sections. In the first section, we evaluate the influence of the algorithm settings on the performance of the models. Next, we have a look at the impact of data preparation and data selection. Finally, we evaluate the overall results and an in-depth analysis is performed on the best performing models.

6.1.1 Algorithm settings impact

If we have a look at the average performance of the models under the different algorithm settings, we find the results visible in Table 16.

Min. records	2	**5**	**2**	**5**
pruning severity	75	**75**	**25**	**25**
Accuracy	0.629	0.622	0.629	0.624
AUC	0.617	0.607	0.626	0.614
Precision	0.643	0.639	0.648	0.643

Table 16: Algorithm settings result

As can be seen in the table, the differences are small (maximum difference of 0.019 or 1.9%). The averages are based on 160 models each (*8 experiments* x *5 folds* x *4 review reasons*) and we see that the models with the *minimum records in the child branch* set to 2 and the *pruning severity* set to 25 perform better on all performance indicators. We would expect that if the *minimum records in the child branch* is set to a low value, a specific tree is built and pruning is required to prevent building a too over-trained model. Therefore, we would expect the settings “2-75” or “5-25” to perform best. However, as shown in Table 16, the results of the settings “2-75” are second best and the differences are only small.

6.1.2 Data preparation and data selection impact

If we analyze the effect of data preparation and data selection, we observe the results shown in Figure 26.

Impact on:	***Accuracy***	**AUC**	**Precision**
Selecting only good exceptions *versus* all exceptions	+3.2%	+ 0.9 %	+ 4.2%
Imputing missing data manually *versus* no imputation	- 1.4%	- 1.0%	- 0.8%
Using pre-feature selection *versus* no feature selection	- 0.3%	+ 0.4%	+ 0.2%

Figure 26: Impacts of data preparation and data selection

Based on these results, we conclude that using only exceptions on which the action is qualified as *good* (cf. Section 2.3.2), results in significant better performances of the models. This indicates that the performance measure successfully selects exceptions from which it should learn. However, as explained in Section 4.4, this could lead to the exclusion of certain situations in the data set.

Imputing missing data manually has a slightly negative effect on the results. Although we know that decision tree algorithms can handle missing data well, we expected that imputing data as performed in Section 4.2 would result in a better performance, since in general we know more about their values. However, these results show the opposite and the algorithms build better models with missing data than manually imputed data.

Using feature selection before the modelling step has negligible effect: the *accuracy* decreases slightly, but the *AUC* and *precision* increase slightly. If the effect is significant, it would be a choice of the management of IBM if they want a model which is less accurate, but which has a better performance in predicting if an action should be taken or not, or the other way around. However, the results indicate that there is barely an effect of using feature selection.

6.1.3 Overall performance

In Appendix K, the full results of the models built during *cross validation* are visible. Based on these results, Table 17 is constructed, in which the average performance and best performance is given. Since the three performance measures together give a good representation of the performance of a model (cf. Section 5.4), the best performing model is determined by the sum of the models' *accuracy, AUC* and *precision*.

RR	**Average**				**Highest**			
	Accuracy	AUC	Precision	S. Dev.	Accuracy	AUC	Precision	S. Dev
R24	0.617	0.621	0.616	0.057	0.635	0.674	0.639	0.081
R25	0.697	0.664	0.719	0.062	0.744	0.747	0.753	0.075
R26	0.550	0.526	0.590	0.060	0.600	0.544	0.662	0.060
R83	0.640	0.652	0.649	0.034	0.666	0.679	0.680	0.036

Table 17: Average- and highest results cross validation

From this table, we observe that the models built for R25 (stocked out) and R83 (projected inventory below MOP) perform best. Although the performance of R25 is higher, the standard deviation of the models is lower for R83, which can be explained by the size of the data sets: the data set of R83 is almost 4 times as big as the data set of R25. The performance should increase significantly before it will be really useful for assisting planners during exception handling, but these review reasons seem to be promising. The results are somewhat worse for R24 (projected stock out), whereas the results for R26 (below MOP) are useless since the best model for this review reason scores a value of 0.544 at the *AUC*, which means that the models built based on the dataset of this review reason and under the settings for the algorithm, are not able to build a model which can distinguish the different classes (*action* and *no action*). If we have a look at the cross validated models built for the best performing experiments, we observe that for R26 only small trees are built compared to other review reasons (Table 18), probably due to the small data set available for this review reason. This explains the low performance of the models for R26, since it determines the class of a record based on at most two features only. Although the decision trees built for R25 are small as well and the data set is about the same size of R26, the models perform much better. This can be explained by the fact that for a stock out situation it is clearer whether an action should be taken or not.

Fold	**R24**		**R25**		**R26**		**R83**	
	Tree depth	features	Tree depth	features	Tree depth	features	Tree depth	features
1	3	7	2	3	2	7	6	18
2	2	5	4	7	2	3	8	14
3	2	7	2	4	2	6	7	23
4	3	7	4	5	2	7	3	11
5	2	3	2	3	2	6	8	17

Table 18: Decision tree information

Since R25 and R83 are most promising, a more in-depth analysis is performed on these two review reasons.

In Table 19, the settings of the best performing models for R25 and R83 are given, which confirm our most interesting finding of Section 6.1.2: only using the exceptions on which the action taken is qualified as *good* are used for training the model and this results in the highest performance.

Review reason	**Minimum number of records in child branch**	**Pruning severity**	**All/good**	**Impute missing data**	**Feature selection**
R25	2	25/75	Good	Yes	Yes
R83	2	25	Good	No	Yes

Table 19: Settings best performing models

Confusion matrices

In Table 20 the confusion matrices of the best performing models are given. Since the goal of IBM is to exclude exceptions on which *no action* is required, we want to maximize the number of correctly predicted *no action* classes. However, there are two errors the model can make: (1) the actual class of an exception is *action,* but the model predicts it as *no action* and (2) the actual class of an exception is *no action,* but the model predicts it as *action*. These two errors should be minimized as much as possible. Since the goal of IBM is to exclude the exceptions on which *no action* is required, minimizing the first error (1) is most important, because a low error rate indicates that if the model predicts an exception as *no action*, it has a high certainty that the actual class is *no action*. The second error (2) should be minimized as well, since it will lead to a higher number of *no action* exception correctly classified. However, the focus should be on the first (1) error. We observe that most errors are made for the classes where an action should be taken, but no action is predicted. This is a result of the small imbalance in the dataset: the *no action* is simply more present in the dataset.

R25		Predicted class	
		No action	Action
Actual class	No action	109	12
	Action	36	31

R83		Predicted class	
		No action	Action
Actual class	No action	368	92
	Action	175	160

Table 20: Confusion matrices

If we analyze the errors made for the first type error based on the action categories explained in Section 2.3.1, the results in Table 21 are found. For every action category the number of errors made related to the category is given. The total number of exceptions related to the action categories is given as well. The percentage gives the number of errors made for a certain action category compared to the total number of exceptions related to a certain action category.

	Forecast		Order		Setting		Other	
	# errors	Total	# errors	Total	# errors	Total	# errors	Total
R25	0 (0%)	23	16 (59.3%)	27	3 (75%)	4	18 (81.8%)	22
R83	10 (14.1%)	71	107 (58.5%)	183	64 (68.1%)	94	15 (44.1%)	34

Table 21: Error action categories

From these results we observe that for actions related to *forecast changes* only a small number of errors is made and these actions seem to be well predictable. For R25 no errors are made for this category, whereas for R83 only a few errors are made. The other three categories seem to be less predictable and a significant number of errors is made for all categories.

Prediction confidence level

Next to the prediction itself, SPSS Modeler provides a confidence level for each prediction as well. If we only consider predictions for which the prediction confidence is above a certain

threshold, we expect that the performance of the model will increase. However, in this case no prediction is made for exceptions with a confidence level below the threshold. This results in a set of exceptions which is better to be predicted by the model and a set of exceptions that should be evaluated by the planners.

In Figure 27 the graphs for R25 and R83 are shown, which give the performance of the model when we vary the prediction confidence threshold. The confidence threshold is shown on the x-axis. In both graphs, we see a blue (*accuracy*) and an orange (*precision*) line, for which the value is shown on the left y-axis. The green line is the average performance of the *precision* and the *accuracy*. The black line represents the fraction of exceptions included in the model for a certain confidence threshold. We observe for both review reasons that the performance indeed is higher when only exceptions with a certain confidence threshold are evaluated by the model, compared to using all exceptions.

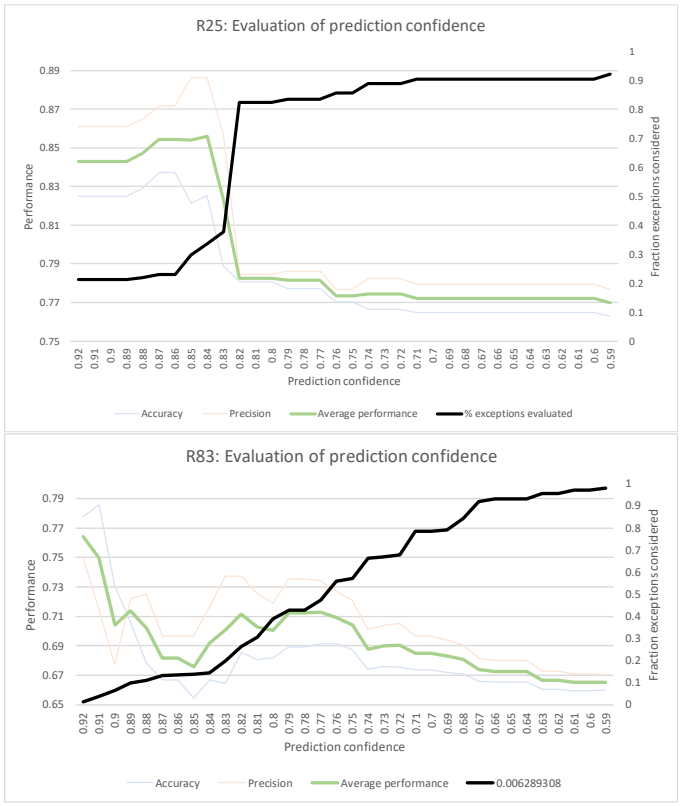

Figure 27: Evaluation of prediction confidence level

Setting a confidence threshold typically results in a higher performance. Setting this value at a somewhat low value, results in including more exceptions in the model with a lower performance, whereas a somewhat higher value results in a higher performance, but less exceptions are included in the model.

For R25 two interesting point are observed: when the prediction confidence threshold is set to 0.84 and when it is set to 0.77. In the first case, the *accuracy* is 82.5% and the *precision* is 88.6%, but only 33.5% of the exceptions is evaluated. In the second case the *accuracy* is 77.7%, the *precision* is 78.6% and 83.5% of the exceptions is evaluated. Compared to the performance without a threshold (*accuracy* of 74.4% and *precision* of 75.3%), these are significant increases.

For R83 the most interesting threshold value is 0.78. In this case, the *accuracy* is 68.9% and the *precision* is 73.5%, where 42.9% of the exceptions is evaluated. Compared to the performance without a threshold (*accuracy* of 66.6% and *precision* of 68.0%), it performs slightly better, but much less exceptions are evaluated.

Decision trees

The decision trees created during *cross validation* for R25 and R83 are shown in Appendix L. From the decision trees we find that for R25 9 distinct features are used and for R83 26, which are shown in Table 22. In this table, the number of times the feature is present in the *cross validated* models is shown and the average effect on the *accuracy, AUC* and *precision* if this feature is removed from the model is given.

R25		
Feature	# present	Effect
Analyzer code	5	-11.0%
# RR	3	-5.1%
Vitality	1	-0.4%
WAC value	1	-0.3%
Return rate	3	-0.2%
Warranty supplier	3	-0.1%
Birth year age	2	0.1%
Forecasted quantity	3	0.3%
Historical activity 1 M	1	0.4%

R83		
Feature	# present	Effect
Analyzer code	5	-3.1%
Birth year age	3	-1.0%
Division owner code	4	-0.6%
Fraction inventory available	3	-0.6%
Shelf life indicator	3	-0.5%
Inventory position severity	1	-0.4%
Open order indicator	1	-0.3%
Stock level	1	-0.3%
Repair supplier	3	-0.2%
Weekend	5	-0.2%
PSS	4	-0.2%
Critical quantity	4	-0.2%
Open order quantity	2	-0.2%
Inventory position	4	-0.2%
Historical activity 1 month	4	-0.2%
Stock level repair	2	-0.1%
Life cycle position	1	-0.1%
# RR	5	-0.1%
Historical activity 2 years	3	0.0%
EOS age	4	0.0%
Return rate	3	0.0%
First stock age	4	0.1%
Minimum lead time	2	0.1%
Vitality	4	0.2%
Forecasted quantity	5	0.2%
Fraction recent demand	4	0.3%

Table 22: Features used in decision trees

If we have a deeper look into the models built for each review reason, we observe that most decision trees first branch on the *analyzer code*, which indicates that this is the most important feature. In Section 4.5 the feature *analyzer code* came out as the most important feature as well. If we exclude this feature from the model, the effects are significant: the *accuracy, AUC* and *precision* reduce on average by 11.0% for R25 and 3.1% for R83. We see a significant effect (a decrease of 5.1%) on the results of R25 if the *# of RR* is excluded as well.

If we have a look at the features used in the models of R25, we observe that only original features are used. The features which are constructed in Section 4.3, which provide information about the *supply chain status*, are not used at all. This is remarkable, because we constructed these features, since we expected these features to bring important information. Next to that, we observe that different sets of features are used per *cross validated* model. This indicates that the training data sets are not stable enough, which could be explained by the small dataset for R25. However, some logical patterns are found in the decision trees. First, we observe that if the *# of RR* is 1, *no action* is predicted and if it is higher than 1, *action* is predicted. This could indicate that if the problem is bigger than the original review reason, an action is required. Second, if the *forecasted quantity* is relatively low, often *no action* is predicted, which could indicate that often *no action* is taken for slow moving parts. A less logical pattern is observed as well: the models predict *no action* if the *vitality* is 1 and *action* if the *vitality* is higher than 1. This is remarkable, since a *vitality* of 1 means that if a part is broken, a machine is down and we would expect that *action* is required for exceptions with a *vitality* of 1.

On the other hand, we observe for R83 that much more features are used and features constructed in Section 4.3 are included in the models as well. Next to that, the *cross validated* models for R83 have a significant number of features in common, which indicates that the training data set is more stable. Also, the trees built for R83 are deeper than the trees built for R25. However, especially for the deeper branches of the trees, we observe some illogical patterns. An example is that *no action* is taken for parts with a *vitality* of 1 where the *critical quantity* is higher than a threshold, whereas *action* is taken if the *critical quantity* is lower than a threshold. The opposite would be expected. Another example is that an *action* is predicted if the *historical demand 1 month* is higher than a threshold and the *forecasted demand* is lower than a threshold, which are interrelated and contradict. These illogical patterns could be a result of the algorithm setting *minimum number of records per child node*, which is set to 2 for the best performing models. A low setting for this setting could result in these illogical branches. On the contrary, a number of logical patterns are found as well. First of all, the same patterns are found for the *# of RR* and the *forecasted quantity* as compared to R25. In contrast to R25, we observe that for parts with a *vitality* of 1, often *action* is taken in these trees. Next, we observe that for young parts (low *life cycle position,* low *birth year age* or low *first stock age*) often an *action* is taken, which could be explained by the fact that demand is hard to predict for relatively young parts. Finally, we observe that if the *fraction inventory position*

available is low or the *fraction recent demand* is high, often *action* is predicted. If we compare the patterns found in the models to the important features found in Section 4.5, we see that the most important features are found in the models and these features show clear patterns, especially for R83.

6.1.4 Results ensemble model

The models built by *cross validation* are combined to one ensemble model and the *training* dataset, which has never been seen by the model yet, is used to determine the performance of the ensemble model. The ensemble model uses the individual votes of the 5 *cross validated* models on each unseen record. The full results are shown in Appendix K.

RR		**Accuracy**	**AUC**	**Precision**
R24	Cross validation	0.635	0.674	0.639
	Ensemble	0.644	0.639	0.59
R25	Cross validation	0.744	0.747	0.753
	Ensemble	0.733	0.75	0.643
R83	Cross validation	0.666	0.679	0.680
	Ensemble	0.644	0.62	0.655

Table 23: Best results cross validation versus ensemble model

In Table 23, we observe that the results of the ensemble models deviate from the results given by the *cross validated* models for the best performing models. If we look at all ensemble models, on average we observe a difference of 3.4% in *accuracy*, 4.7% in *AUC* and 3.8% in *precision* between the *cross validated* models and the *ensemble* model. The average *accuracy* itself is for the ensemble model -0.4%, the *AUC* -1.8% and the *precision* -1.2%, compared to the *cross validated* models. These numbers indicate that the ensemble models do not perform better compared to the *cross validated models.* This is probably the result of the limited size of the datasets, which results in unequal folds and sets of data, regarding the exception situations within a set. However, we would expect that if *cross validated* models are combined, this would result in a more accurate model.

6.2 Comparison to *logic derived* rule set

In this section, a simple decision tree is built, based on the findings in Section 2.3.2 and Section 6.1.3, to see whether machine learning algorithms are of added value. The performance of this decision tree is evaluated on the datasets for which the highest performance is achieved.

In section 2.3.2. we observed that in most cases *no action* is taken on exceptions of parts for which the *hub is not responsible* or for which the parts *are not stocked*. Next to that, on slow movers often *no action* is taken, which was observed by a lower average *historical demand* on exceptions on which *no action* is taken. The threshold is set to 7.5, since the boundaries of the mean plus (*no action*) or minus (*action*) a certain standard deviation meet at the threshold.

Next to the observed statistics in section 3.2.3, we observed in the models that for exceptions where only one review reason is connected to the exception, often *no action* is taken. More than one review reason connected to an exception often indicates that an *action* should be taken. Also, we observed that often *no action* is taken if the *fraction recent demand* is low. On average this threshold is 0.34. Finally, the *fraction inventory position available* indicates that *action* is taken if this fraction becomes below a certain value. Based on the average value of the splits until the second level of the trees, this value is set to 0.188.

Based on these findings, we built a simple set of rules:

- If the hub is not responsible -> *no action*
- If the part is not stocked -> *no action*
- If the number of review reasons is 1 -> *no action*
- If the number of review reasons is 2 or more -> *action*
- If the *fraction recent demand* is lower than 0.34 -> *no action*
- If the *fraction recent demand* is higher than 0.34 -> *action*
- If the *fraction inventory position available* is lower than 0.188 -> *action*
- If the *fraction inventory position available* is higher than 0.188 -> *no action*
- If the demand last period is lower or equal to 7.5 -> *no action*
- If the demand last period is higher or equal to 7.5 -> *action*

If these rules contradict, the most predicted class is given to the exception. However, if both classes are predicted equally, the predicted class becomes *action*. These rules are applied to the best performing algorithms. However, since feature selection is applied for these experiments, the feature *hub is not responsible* and *part is not stocked* are excluded from the datasets. Next to that, due to feature selection, the feature *fraction inventory position available* is not present in the dataset of R25. The rules lead to the following results:

R25		Predicted class	
		No action	action
Actual class	No action	112	9
	action	61	6
Accuracy		62.8%	
Precision		64.7%	

R83		Predicted class	
		No action	action
Actual class	No action	406	54
	Action	289	46
Accuracy		56.9%	
Precision		58.4%	

Table 24: Confusion matrices "logic derived rules"

We observe that, compared to the results of the *cross validated* models, these simple rules lead to a much lower *accuracy* and *precision*. On top of that, the model is less able to

distinguish between classes. Therefore, we conclude that machine learning algorithms can produce better results.

6.3 Conclusion

In this chapter, the evaluation step of the CRISP-DM methodology is performed, which evaluates the results of the models. The following conclusions are found:

- Experiments with the algorithm settings do not give significantly different results, but the best performance is found if the *minimum records per child branch* is set to 2 and the *pruning severity* is set to 75.

- Using only exceptions on which the decisions are qualified as *good* increases the performance significantly, whereas imputing missing data manually decreases the average performance. Surprisingly, for the best performing models, missing data is manually imputed. Using feature selection before the modelling step does not lead to significantly different results.

- R25 and R83 are most promising for assisting planners during the exception handling. However, the limited size of the data sets have an impact on the results, which are quite unstable. For the models built for R25 and R83:

    - Most mistakes are made where *no action* is predicted, whereas an *action* should be taken. For exceptions of the action category *forecast related* few mistakes are made, whereas for the other categories (*order related, setting related* and *other*) many mistakes are made.
    - Setting a confidence threshold for the predictions leads to a higher performance, but it decreases the number of exceptions handled by the model.
    - The feature *analyzer code* seems to be the most important feature for both models. The models built for R25 are less stable than the models built for R83. In the models of both review reasons, patterns are found which make sense.

- Ensemble models perform worse than the *cross validated* models.

- Machine learning algorithms have a beneficial effect compared to logic rules deduced from data analysis and model analysis.

Chapter 7: Deployment

From the previous chapter, we have found that the models of R25 and R83 are most promising to use as assistance during the exception handling. The models constructed by the algorithms for these two review reasons are deployed, which is the focus of this chapter. Next to that, a plan of monitoring and maintaining the tool is introduced. The sixth step of the CRISP-DM is the focus of this chapter, as can be seen in Figure 21.

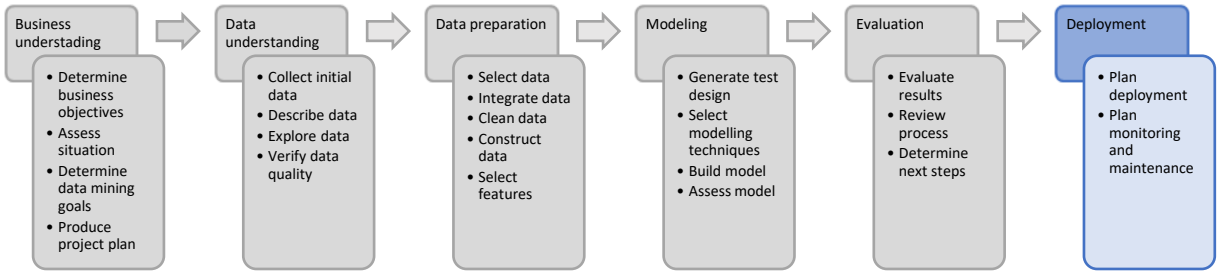

Figure 28: Current focus within CRISP-DM methodology (Chapman et al., 2000)

7.1 Plan deployment

Since the performance and the models of R25 and R83 are promising, a prototype is built for these review reasons based on the models. This prototype is implemented in Excel combined with SPSS Modeler. A planner fills in characteristics of the part to which the exception is related in an Excel file, saves and closes the file, and runs the model in SPSS Modeler. SPSS Modeler shows whether an action should be performed or not. For R25 9 features are used and for R83 26 features. These features can be found in Table 22 in section 6.1.

Since filling in the data and running the model is quite intensive, the possibilities of implementing such a solution in *Servigistics* should be considered. In particular, this step may be valuable if future research may realize further performance improvements and if, in a next step, it may become possible to predict action types as well.

In any case, we suggest IBM to first train a model that is fed with more data to get a more accurate model, since we observed variations in results, especially for the review reasons where the dataset is smaller. We observed a much lower standard deviation for the models built for R83, compared to the other review reasons. The dataset of R83 contained in some cases more than four times the number of records compared to the other review reasons. To further decrease the standard deviation and therefore increasing the stability of the models, it is suggested to use data of at least one year. This could also lead to a higher performance.

7.2 Plan monitoring and maintenance

If the tool is implemented in the work environment of the planners, the tool should be used carefully. Since the models are built on historical data, the correctness of the tool should be checked by their users in the first place.

Next to that, the models should be rebuilt periodically, since more and possibly different data becomes available, which could result in a different and/or better model. If the model is not rebuilt from time to time, the *accuracy* of the model will drop drastically if circumstances have changed. Therefore, we suggest IBM to use the tool carefully. Next to that, it may be valuable to pay attention to exceptions on which a different action has been taken than predicted and especially pay attention to the reasoning why a planner diverges from the predicted class by the model. The planners should be able to provide important information on how and why they deviate from the recommendation of the tool. Based on this information important changes can made to the data fed to the model or the model itself.

Conclusion, limitations and future research

In this final chapter, the research is concluded. Furthermore, the limitations are discussed and recommendations for future research are given.

Conclusion

This research focused on the application of machine learning during the exception handling process and resulted in the following research problem:

How can machine learning techniques be effectively applied on different review reasons to improve the efficiency and effectiveness of the exception handling process?

Since Schultz' research resulted in an accuracy of only 59%, the following choices are made to find an answer on our research problem:

- The CRISP-DM methodology is followed for this research
- The classification categories are simplified to whether an action is taken or not
- Since machine learning algorithms can only learn from good examples, a performance measure is introduced to qualify decisions made by planners.
- Planners' expertise is incorporated in this research to determine valuable information
- Experiments are performed with different settings and approaches to determine the best approach

This approach resulted in a model built in SPSS Modeler 18.1, which automatically merges all data related to exceptions into a dataset and performs different operations to this dataset. Since a clean dataset is essential to effectively apply machine learning algorithms, most time should be and is spent to the data exploration and data preparation steps of the CRISP-DM methodology.

Based on experiments, the best performing algorithm is found to be the C5.0 algorithm. The best performing models are achieved if the *minimum number of records in child node* set to 2, the *pruning severity* set to 25, only exceptions on which the actions taken are qualified as *good* are used and feature selection is applied before the modelling step. Using *cross validation* this resulted in an *accuracy* of 74.4%, an *AUC* of 74.7% and a *precision* of 75.3% for R25 and an *accuracy* of 66.6%, an *AUC* of 67.9% and a *precision* of 68.0% for R83, which is promising. However, the consistency of R83 is much higher compared to R25, which is probably a result of the differences in the size of the datasets. The results for R24 were lower and the results of R26 are not taken into account, since the models were not able to build appropriate models, possibly due to the limited size of the data set. Although R25 has about the same data set size, it performs well, possibly due to the fact that whether an action should be taken on a current stock out situation is clearer.

If we have a deeper look into the errors made, we see that primarily the action categories *order related, setting related* and *other* are wrongly predicted, whereas the *forecast related* actions often are well predicted.

The results for R25 and R83 can be further improved by using a threshold for the prediction confidence, however not all exceptions are in this case evaluated. For R26 an *accuracy* of 82.5% and a *precision* of 88.6% can be achieved, where 33.5% of the exceptions is considered. For R83 this is 68.9% and 73.5%, where 42.9% of the exceptions is considered.

If we have a look at the models built by the algorithm, we observe that deeper models are built and more features are used in the models for R83, compared to R25. On top of this, for the different folds of R83 often a similar set of features is used, which indicates that the models are more stable, which is in a lesser extent observed for R25. These observations are probably a result of the bigger data set available for this review reason. For both models, patterns are discovered which make sense.

The ensemble models perform worse than the *cross validated* models and on top of that, the results deviate significantly from the *cross validated* results. This is probably caused by the limited size of the datasets which results in unequal datasets, regarding exception situations within a dataset.

If we compare these results with the results of Schultz (2017), we observe that we used a structured approach, which resulted in a less complex model with a higher *accuracy*. Simplifying the approach was necessary to examine the possibilities of using machine learning algorithms during the exception handling process.

The results for R25 and R83 have both strong sides (R83 is more consistent, the models built are intuitively better, whereas the performance of R25 is better), which indicate that using machine learning for the exception handling is promising, but the performance is simply not high enough to be a true planner assistant. Especially the size of the datasets seems to have quite an impact on the models built by the algorithms and their performance. Therefore, we suggest IBM to continue gathering data and use the same approach as we did to see whether the performance can be increased and be more consistent.

Limitations

Although this research gave some interesting results, some limitations should be addressed.

Since the planners operate from outside Europe, it was not possible to get familiar with the exception handling process by working with the planners. Although team leaders and employees who performed the job as planner in the past were available, this sometimes led

to contradictory information. It would be valuable to see the planners working on a day to day business. Exception handling is a complex process and expertise is needed.

Most analysis is therefore performed on available data. However, the retrieval of historical data is not possible, which meant that data should be collected during the assignment. Since time is limited, this led to a very limited dataset. Nowadays, algorithms are able to analyze big amounts of data in a couple of minutes. The biggest dataset we had, contained less than 2.000 records. To draw conclusions on such a small dataset, is risky.

Next to the small dataset, it is not known whether all adjustments made by planners are recorded in *Servigistics*, which is a major drawback of this research. It could well be that a planner communicates by making a call or sending an e-mail and these *adjustments* are not visible in the data, since these adjustments are not recorded in *Servigistics* and thus are not taken into account.

Future research

Although the results for R26 are barely better than guessing, the same approach led to promising results for R25 and R83. To truly assist planners during the exception handling, future research should be considered. In the first place, we would recommend IBM to collect data over a longer period, which would lead to a bigger dataset. Building a model on more data probably leads to more accurate and stable results. Since a lot of different situations exist which trigger exceptions, collecting more exceptions also gives a more complete representation of reality.

Next to that, we recommend involving the planners in India in two ways. First, they can provide important information about the aspects on which their decision is based and therefore, important features can be constructed and included in the model. Secondly, the planners could be involved by providing feedback on why the recommended action by an algorithm is a good/bad one, which could improve models.

Furthermore, we recommend considering online learning algorithms as a next step, in which the exceptions, their circumstances and the action taken by planners are tracked automatically and the model is updated after every new exception. A lot of data is available in the systems of IBM, which could be included in an online learning algorithm. On the longer term, we also recommend incorporating the different action categories in the models. It could be interesting to see in what extent it is able to predict the right action category.

Finally, we focused in this research on how an exception is solved. However, we did not have a look at the settings of triggering the exceptions. Another interesting future research could focus on optimizing the exception triggering process. Possibly different exception triggering settings lead to less exceptions and less critical situations arise.

References

Andrew, A. (2012, 21 July). Why split data in the ratio 70:30? [Online article]. 12 januari 2019, http://information-gain.blogspot.com/2012/07/why-split-data-in-ratio-7030.html

Azevedo, A., & Santos, M. (2008). *KDD, SEMMA AND CRISP-DM: A PARALLEL OVERVIEW*. http://recipp.ipp.pt/bitstream/10400.22/136/3/KDD-CRISP-SEMMA.pdf

Baguley, T. (2012). *Dealing with missing data. Online Supplement 2 to Serious stats: A guide to advanced statistics for the behavioral sciences*. https://www.researchgate.net/publication/312085497_Dealing_with_missing_data_Online_Supplement_2_to_Serious_stats_A_guide_to_advanced_statistics_for_the_behavioral_sciences_Basingstoke_Palgrave

Bell, J. (2014). Machine learning: Hands-On for developers and technical professionals. Wiley.

Brownlee, J. (2016, 7 June). 8 Tactics to Combat Imbalanced Classes in Your Machine Learning Dataset [Online article]. 26 Octobre 2018, https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/

Cavalieri, S., Garetti, M., Macchi, M. & Pinto, R. (2008). A decision-making framework for managing spare parts, Production Planning & Control: The Management of Operations, 19:4, 379-396, DOI: 10.1080/09537280802034471

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. (2000). *CRISP-DM 1.0*. https://www.the-modeling-agency.com/crisp-dm.pdf

Gelman, A. (2006). *25: Missing-data imputation*. http://www.stat.columbia.edu/~gelman/arm/missing.pdf

Gupta, B., Rawat, A., Jain, A., Arora, A., & Dhami, N. (2017). Analysis of Various Decision Tree Algorithms for Classification in Data Mining. *International Journal of Computer Application*, *163*, –268. https://pdfs.semanticscholar.org/fd39/e1fa85e5b3fd2b0d000230f6f8bc9dc694ae.pdf

Gutierrez-Osuna, R. (n.d.). Sequential feature selection [Lecture slides]. 12 January 2019, http://research.cs.tamu.edu/prism/lectures/pr/pr_l11.pdf

IBM. (n.d.-a). IBM100 - IBM Is Founded. 26 November 2018, https://www.ibm.com/ibm/history/ibm100/us/en/icons/founded/

IBM. (n.d.-b). Data Science and Machine Learning. 19 September 2018 https://www.ibm.com/analytics/machine-learning

IBM. (n.d.-c). IBM SPSS Modeler 18.1.1 Modeling Nodes. 02 February 2019 ftp://public.dhe.ibm.com/software/analytics/spss/documentation/modeler/18.1.1/en/ModelerModelingNodes.pdf

IBM. (2005). IBM SPO Service Parts Operations EMEA [Presentation]. 26 September 2018, http://www.ibm.com

IBM. (2014, 28 januari). *IBM Service Parts Operations (SPO): The right part in the right place at the right time* [Video]. 12 October 2018, https://www.youtube.com/watch?time_continue=11&v=EOKrABwiaFc

Jha, V. (n.d.). Machine Learning Algorithm - Backbone of emerging technologies. 19 September 2018, https://www.techleer.com/articles/203-machine-learning-algorithm-backbone-of-emerging-technologies/

Jovic, A., Brkic, K., & Bogunovic, N. (2015). A review of feature selection methods with applications. *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, https://doi.org/10.1109/mipro.2015.7160458

Koopman, C. (2011). Optimizing the last time buy decision at the IBM Service Part Operation organization.

Kotsiantis, S. (2007). Supervised Machine Learning: A Review of Classification Techniques. *Informatica*, *31*, 249–268. https://datajobs.com/data-science-repo/Supervised-Learning-[SB-Kotsiantis].pdf

Kuhn, M., & Johnson, K. (2018). *Applied Predictive Modeling*. New York, United States: Springer New York.

Kwak, S. K., & Kim, J. H. (2017). Statistical data preparation: management of missing values and outliers. *Korean Journal of Anesthesiology*, *70*(4), 407. https://doi.org/10.4097/kjae.2017.70.4.407

Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., & Liu, H. (2017). Feature Selection. *ACM Computing Surveys*, *50*(6), 1–45. https://doi.org/10.1145/3136625

Liu, H., Motoda, H., & Zhao, Z. (2010). Feature selection: An ever evolving frontier in data mining. *The Fourth Workshop on Feature Selection in Data Mining*, *4*, 407.

Moncrief, E. C., Schroder, R. M., & Reynolds, M. P. (2006). *Production Spare Parts: Optimizing the MRO Inventory Asset*. Industrial Press.

Pereira, J. M., Basto, M., & Da Silva, A. F. (2016). The Logistic Lasso and Ridge Regression in Predicting Corporate Failure. *Procedia Economics and Finance*, *39*, 634–641. https://doi.org/10.1016/s2212-5671(16)30310-0

Schultz. (2017). *Using machine-learning models for operational exception handling*. https://essay.utwente.nl/73979/1/Schultz_MA_BMS.pdf

Srinidhi, S. (2018, 29 juli). How to split your dataset to train and test datasets using SciKit Learn [Online article]. 12 January 2019, https://medium.com/@contactsunny/how-to-split-your-dataset-to-train-and-test-datasets-using-scikit-learn-e7cf6eb5e0d

Sunasra, M. (2017, 11 November). Performance Metrics for Classification problems in Machine Learning [Online article]. 26 oktober 2018, https://medium.com/greyatom/performance-metrics-for-classification-problems-in-machine-learning-part-i-b085d432082b

Tan, P. N., Steinbach, M., & Kumar, V. (2006). *Introduction to Data Mining* (5e ed.). Boston, United States: Pearson Education, inc..

Visalakshi, S., & Radha, V. (2014). A literature review of feature selection techniques and applications: Review of feature selection in data mining. *2014 IEEE International Conference on Computational Intelligence and Computing Research*, https://doi.org/10.1109/iccic.2014.7238499

Wolpert, D., & Macready, W. (1997). No Free Lunch Theorems for Optimization. *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION,*, *1*(1), 67–82. https://ti.arc.nasa.gov/m/profile/dhw/papers/78.pdf

Xelus. (2006). *XelusPlan User Guide* (5e ed.).

Appendices

Appendix A

This appendix contains a list of the active review reasons. Data is gathered for a period of one year, from 01-09-2017 – 31-08-2018. A total of 39.701 review reasons occurred in this period and per review reason the number of corresponding exceptions is showed. The list is sorted on the number of exceptions, descending, and the adjusted priority is given.

RR	Name	# of occurrence	% of occurrence	Priority
R38	Order Increase Outside L/T	7986	20,1%	8
R83	Projected Inventory Below MOP	7709	19,4%	17
R80	Projected Inventory Above XS Point	4590	11,6%	11
R81	Supply Constraint Over Consumed	3296	8,3%	20
R24	Projected Stock Out	3270	8,2%	16
R25	Stocked Out	3085	7,8%	14
R26	Below Must Order Point	1765	4,4%	15
R4	Backorder Quantity	1659	4,2%	2
R17	History Value Out of Range	1436	3,6%	4
R9	Behind Sched. Over Policy	1390	3,5%	12
R34	Failed to Auto Approve Orders	1044	2,6%	5
R78	Order Requires Legal Entity Approval	948	2,4%	21
R41	Quantity Outside Loc/Hier	409	1,0%	18
R68	Returned Stock Overused	289	0,7%	19
R15	Change Forecast Setting	283	0,7%	3
R2	Forced Worksheet	269	0,7%	1
R39	Order Decrease Outside L/T	143	0,4%	9
R37	Unsatisfied Allocation Rqmt	119	0,3%	10
R5	Order Decrease Inside L/T	5	0,0%	7
R7	Order Increase Inside L/T	3	0,0%	6
R19	Order on Sumcode 2 Part	3	0,0%	13

Table 25: Active review reasons. Descending on "# of occurrence".

Appendix B

This appendix contains a table with the explanation of the review reasons we focus on. For explanation of the other review reason the report of Schultz (2017).

Review reason	Description
R24: Projected Stock Out	A stock out is projected within the entire future horizon and the order recommendations are constrained such that the stock out cannot be avoided. All three of the following conditions must be met to trigger this review reason: • The current on hand balance, calculated over the prime/alternate chain must be greater than zero. • The sum of the total requirements from the current period through the end of the planning horizon must be greater than zero. • The projected inventory calculated of the prime/alternate chain must be negative on one or more days in the future.
R25: Stocked Out	The *onhand balance* for the item is equal to zero and the sum of total requirements minus any shortfall is greater than zero. The *onhand balance* is calculated over the prime/alternate chain and the total requirements are summed over the next two years.
R26: Below Must Order Point	The *onhand balance* for the item is below the *must order point* in the current period. The followingen conditions must be met: • The *onhand balance* is a nonzero quantity. • Must order point is a nonzero quantity. If the *onhand balance* is zero, R25 is triggered instead.
R83: Projected Inventory Below MOP	The inventory for the part in the network goes below the must order point within the entire future horizon, because one or more ordering constraints are preventing orders from being recommended. Order constraints may be one or more of the following: • No valid sources. • “Good as it gets” dates set. • Capacity or supply constraints.

Table 26: Explanation of (selected) review reasons

Appendix C

This appendix contains a table with the gathered data from the *Servigistics* databases.

Data	Explanation	Data type
# of review reasons	If an exception is triggered, this feature gives the number of review reasons attached to this exception.	Discrete
A code	Code which represents the order method code.	Nominal
Air Support Indicator	Indicator which specifies whether a part is stocked near an airport.	Binary
Analyzer code	Code which specifies the part category, part brand and planner.	Nominal
Auto-order indicator	Indicator which specifies whether auto-ordering is on for at least one order type.	Binary
B supplier	Indicator whether a NBO supplier is available.	Binary
Birth date	Date when the part was first introduced in the system of IBM.	Discrete
C code	Code which represents where it is recommended to stock the part.	Nominal
CI144 MFGXFER indicator	Indicator which specifies whether the part is in the process of being transferred from Manufacturing source responsibility to part logistics source responsibility.	Binary
Critical quantity	Quantity which is the number of pieces deemed to be critical in the supply versus need of the part.	Discrete
Critical indicator	Indicator which specifies whether a part is critical.	Binary
Critical part unsatisfied indicator	Code which specifies whether the status and activities on part in a critical supply situation are deemed to be unsatisfactory by the planner.	Binary
Division group name	Identifier of a group of division codes.	Nominal
Division owner code	Code which specifies the product platform associated with the invoice booking.	Nominal
End of service date	Date which indicates that the part is not supported anymore.	Discrete
First stock date	Date when a part was first stocked.	Discrete
Forecast method	Forecast method which is used for planning the part.	Nominal
Forecasted quantity	Demand forecast for the coming 26 periods (4 weeks per period).	Continuous
G supplier	Indicator whether a GARS supplier is available.	Binary
Historical activity 1 period	Actual demand in the last period.	Discrete
Historical activity 2 years	Actual demand in the last 26 periods.	Discrete
Hub responsible indicator	Indicator which specifies whether the hub is responsible for the part.	Binary
Last Time Buy indicator	Indicator which specifies whether a last time buy process was performed.	Binary

Minimum lead time	The minimum lead time for a part in weeks.	Discrete
Material Class	Indicator which represents a combination of value of the part and demand of the part (A = high value, F = low value).	Nominal
NBO indicator	Indicator which specifies whether the part is a new business opportunity part.	Binary
New buy supplier	Indicator whether a new buy supplier is available.	Binary
New part indicator	Indicator which specifies whether the part is a recently introduced part.	Binary
Open orders quantity	Quantity of open orders.	Discrete
Open order arrival	Number of weeks before an open order arrives	Continuous
Policy safety stock level	The PSS is used as buffer to deal with uncertainty in demand.	Discrete
Repair supplier	Indicator whether a repair supplier is available.	Binary
Return rate	Rate which specifies the fraction of parts that return unused.	Continuous
Shelf life	The shelf life of a part in years.	Continuous
Stock level / inventory level	Current on hand balance in the network/hub.	Discrete
Stock repair	Current on hand repair stock (available for repair)	Discrete
Stock warranty	Current on hand warranty stock (available for warranty)	Discrete
Stocked indicator	Indicator which specifies whether it is allowed to stock a part at the Hub.	Binary
Successor	Indicator which specifies whether a part has a successor.	Binary
Usage	A categorical value which represents the historical usage of the part (1 = low usage, 5 = high usage).	Nominal
Vitality	Indicator which specifies the vitality of a part (1 = vital part, 5 = non-vital part).	Ordinal
WAC	Weighted average cost: value of a part. This is an average value of the parts on stock, since prices can vary.	Continuous
Warranty supplier	Indicator whether a warranty supplier is available (providing warranty parts)	Binary
Weekend	Indicator whether the exception was triggered during the weekend or during weekdays. Some information is only updated in weekends.	Binary

Table 27: Overview of features and their explanation

Appendix D

This appendix contains the queries used for gathering data from different databases.

Vitality data

```
Select PARTnumb, PART_VIT, INIT_UNITCOST_USD, NBO_IND
from ogsdbap.OGSRTA0_PART
where substr(partnumb,1,5) in ('00000','03003')
```

Successor data

```
Select partnumb, related_partnumb
from ogsdbap.OGSRTA1_PART_REL
where relatn_type not in ('SK','UN') and prime_succsr_ind = 'S'
```

Return Rate

```
SELECT ITEM_ID
      ,SUM(CASE WHEN FIN_TRX_TYP = 'K1' THEN (TRX_PARTS_QTY * -1)
                                        ELSE 0
           END)  SHIPPED_QTY
      ,SUM(CASE WHEN FIN_TRX_TYP = 'K2' THEN TRX_PARTS_QTY ELSE 0
           END)  GOODRET_QTY
FROM COPPC.PDWIC_EVENT_HNDLR
WHERE SUPLY_AVAIL_CAT  = 'NEW'
  AND TRX_DATE         > CURRENT DATE - 732 DAYS

GROUP BY ITEM_ID
```

Part planning data

```
select ptb1.partnumb, sum(ptb1.reorder_point) reorder_point, sum(ptb1.order_upto_point) order_upto_point,
       sum(ptb1.critical_stock_lvl) critical_stock_lvl, sum(ptb1.exc_stock_lvl) exc_stock_lvl
from ogsdbap.OGSPTB1_STOCK_PLAN PTB1, ogsdbap.OGSrtb8_stock_loc RTB8,OGSDBAP.OGSRTB2_COUNTRY RTB2

WHERE PTB1.STOCK_LOC_ID = RTB8.STOCK_LOC_ID AND RTB8.ISO_CNTRY_COD = RTB2.ISO_CNTRY_COD
AND (SUBSTR(RTB2.IBM_REGION_ID,1,4) = 'EMEA' or rtb2.iso_cntry_cod = 'CN')
and ptb1.stock_pln_type in ('LT','AT')

group by ptb1.partnumb

with ur;
```

Part data

```
Select partnumb, analyzer_cod, part_wac, part_wac_curr, div_owner_cod,
air_support_ind, hub_rsp_ind, nbo_ind, birth_date, first_stock_date,
eos_date, eol_xfer_ind, ci144_mfgxfer_ind, lasttime_buy_ind, c_and_a_cod,
stocked_ind, clc_ind, local_owner_cod, new_part_ind, critpart_ind, critpart_usat_ind
from ogsdbap.OGSRTA6_PART_LOC
where STOCK_LOC_ID = '788-5000'
```

Prime – alternate data

```
select distinct prime, alternate from(
SELECT VALUE(RTA1.RELATED_PARTNUMB,RTA0.PARTNUMB) AS PRIME,
       VALUE(RTA1_2.PARTNUMB,RTA0.PARTNUMB)       AS ALTERNATE

FROM OGSDBAP.OGSRTA0_PART     RTA0
  LEFT OUTER JOIN
     OGSDBAP.OGSRTA1_PART_REL RTA1
  ON  RTA0.PARTNUMB          = RTA1.PARTNUMB
  AND RTA1.PRIME_SUCCSR_IND  = 'A'
  AND RTA1.RELATN_TYPE      <> 'SK'
  LEFT OUTER JOIN
     OGSDBAP.OGSRTA1_PART_REL RTA1_2
  ON (RTA1.RELATED_PARTNUMB   = RTA1_2.RELATED_PARTNUMB
  AND RTA1_2.PRIME_SUCCSR_IND = 'A'
  AND RTA1_2.RELATN_TYPE      <> 'SK')
  OR RTA1.RELATED_PARTNUMB    = RTA1_2.PARTNUMB) a

where prime <> alternate

WITH UR;
```

Material class data

```
select ptd1.partnumb, CHAR(PTD1.PARM_VALUE || PTD1A.PARM_VALUE,3) AS MAT_CLASS

FROM OGSDBAP.OGSrta6_PART_loc rta6,OGSDBAP.OGSPTD1_PLN_PARMDT PTD1,OGSDBAP.OGSPTD1_PLN_PARMDT PTD1A

WHERE PTD1.PARTNUMB = PTD1A.PARTNUMB
and ptd1.partnumb = rta6.partnumb
AND PTD1.STOCK_LOC_ID = PTD1A.STOCK_LOC_ID
AND rta6.stock_loc_id = ptd1.stock_loc_id
AND PTD1A.STOCK_LOC_ID = '788-5000'
AND PTD1.PARM_ID = 'ABCCode'
AND PTD1A.PARM_ID = 'QtyCode'
and rta6.div_owner_type = 'D'

with ur;
```

Inventory data

```
Select PARTNUMB, ptb0.stock_loc_id, STOCK_INVENT_TYPE, ACTUAL_ONHAND_BAL
from ogsdbap.OGSPTB0_STOCK_INV PTB0,ogsdbap.OGSrtb8_stock_loc RTB8,OGSDBAP.OGSRTB2_COUNTRY RTB2

WHERE PTB0.STOCK_LOC_ID = RTB8.STOCK_LOC_ID AND RTB8.ISO_CNTRY_COD = RTB2.ISO_CNTRY_COD
AND (SUBSTR(RTB2.IBM_REGION_ID,1,4) = 'EMEA' or rtb2.iso_cntry_cod = 'CN'
or substr(ptb0.stock_loc_id,1,1) = 'S')
```

Historical activity data

```
SELECT PTB3.PARTNUMB, PTB3.HIS_ACT_PERIOD, PTB3.HIS_ACT_TYPE, sum(ptb3.his_act_qty) his_act_qty
FROM OGSDBAP.OGSPTB3_HIST_ACT PTB3

INNER JOIN OGSDBAP.OGSRTB8_STOCK_LOC RTB8
ON PTB3.STOCK_LOC_ID = RTB8.STOCK_LOC_ID

INNER JOIN OGSDBAP.OGSRTB2_COUNTRY RTB2
ON RTB8.ISO_CNTRY_COD = RTB2.ISO_CNTRY_COD

INNER JOIN OGSDBAP.OGSRTB1_IBM_REG RTB1
ON RTB2.IBM_REGION_ID = RTB1.IBM_REGION_ID

where (RTB1.IBM_GEO_ID = 'EMEA' or substr(ptb3.stock_loc_id,1,1) = 'S' or rtb2.iso_cntry_cod = 'CN')
GROUP BY PTB3.PARTNUMB, PTB3.HIS_ACT_PERIOD, PTB3.HIS_ACT_TYPE
```

Forecast data

```
Select *
from ogsdbap.OGSPTB4_FORECAST
where STOCK_LOC_ID = '788-5000' and fcst_type = 'FcstCalc'
```

Critical part data

```
Select *
from ogsdbap.OGSPTC4_CRIT_PARTS
where STOCK_LOC_ID = '788-5000' and obs_ind = 'N'
```

Contract info data

```
Select a.PARTNUMB, a.ord_loc_id, a.SUPL_ORDER_TYPE, a.PART_PRICE, a.PART_PRICE_CURR,
a.LEAD_TIME, a.ORDSCEN, a.MIN_ORDER_QTY, a.SRC_PRIO, b.GAIG_DATE
from ogsdbap.OGSRTC2_CONTRA_INF a

left outer join ogsdbap.ogsrtc4_part_src b
on a.CONTRA_INFO_NUM = b.CONTRA_INFO_NUM
and a.partnumb = b.partnumb
and a.ord_loc_id = b.ord_loc_id

where substr(a.ORD_LOC_ID,9,4) <> '-000'
and a.ordscen in ('HUB','LOC')
```

Open orders & auto-order data

For these two queries we refer to the supplementary file, since they are too extensive to include in the report.

Appendix E

In this appendix, the data model built in SPSS Modeler is explained. As can be seen in Figure 29 this model consists of 15 nodes. The 15 nodes are explained in consecutive order, based on this figure. The node *RR+ManualAdjustments*, which represents the exceptions triggered and the action taken, is the input. In the consecutive loop, data related to the status of the part is merged and the node *FinalModelOutput* represents the dataset used for further study. Since data is gathered from different databases, data is merged from 15 nodes to one single file. The data of each feature of the exceptions is merged based on the week number and the part number.

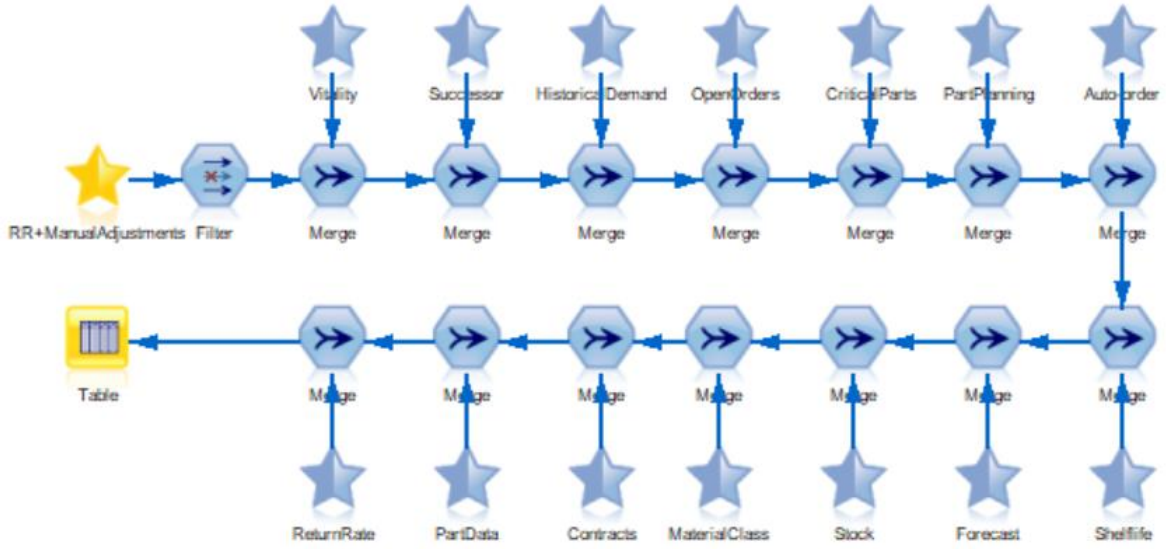

Figure 29: Data integration: merging data to exceptions

As explained in 2.3.1, review reasons are triggered on the parts which are planned, the so-called *prime parts*. This has the consequence that for many of the following nodes, *alternate* part numbers are translated to the *prime* part number. For example: the historical demand of a prime part is the sum of the historical demand of the *prime* and *alternate* parts. However, since data analysis showed that it is not always the case that an exception is triggered on the prime part, also the original data is kept.

RR+ManualAdjustments

In this node, which can be seen in Figure 30, Figure 31 and Figure 32, the review reasons are evaluated, adjustments are connected to the review reasons and the performance of decisions is evaluated. This node is split into three parts: *adjustments, review reasons* and *data set creation*.

The *adjustments* are gathered from the PTITXLUA log. In this log, all manual adjustments made worldwide are collected. Since the scope of the research is only the EMEA-region, only the adjustments made by the analyzers working for EMEA are considered. This is done in the six

steps of the lower left corner of Figure 30. Since review reasons are primarily triggered on the prime-part, the adjustments are connected to the prime parts. In this way, a dataset is created which consists of adjustments made for the EMEA-region connected to the alternate- as well as the prime-part.

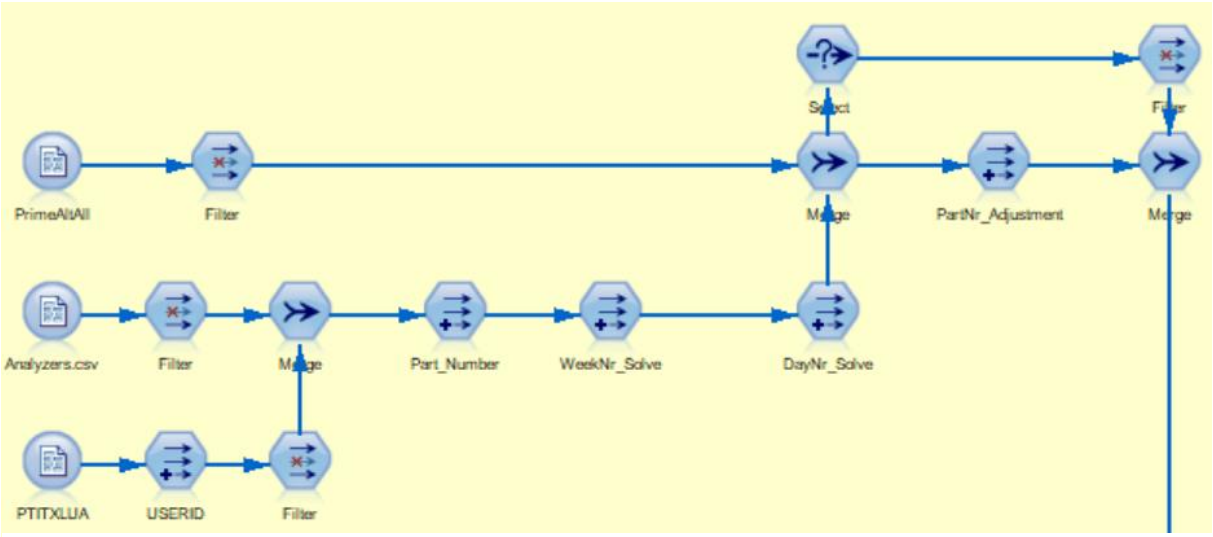

Figure 30: Adjustments

The *review reasons* are gathered from the 3TA1 table. In this table, information about the review reasons (like the part number, trigger date, resolve date) is given. This table is used and transformed to connect adjustments to the exceptions. In the first four nodes, some data is reformatted: the *review reason ID, division group name* and *date*. In the *aggregate* node, the number of review reasons connected to an exception is calculated. In the *select* node, only the chosen review reasons to focus on are filtered. Since there were some inconsistencies in the timestamps of the derived 3TA1 table, the *approval date* and *resolve date* are adjusted. The logic behind these modifications is that the code for creating this table assumes that no work is performed in weekends and data showed this is not true. Next some extra information is derived from data. The *decision period* determines the date on which the planner took an initial decision and follows the following rule:

1. If the exception is not approved, but it is resolved, the *decision period* is the resolve date.
2. If the exception is approved and not resolved, the *decision period* is the trigger date + 7 days[2].
3. If the exception is approved within the number of disabling days and the exception is not yet resolved, the *decision period* is the approval date.
4. If the exception is approved and the exception is resolved within 7 days from the trigger date, the *decision period* is the resolve date.
5. If the exception is approved outside the number of disabling days, then the *decision period* is the trigger date + 7 days[2].

[2] These 7 days follow from the fact that a planner is supposed to take action within a week.

Finally, the review reasons get an ID number.

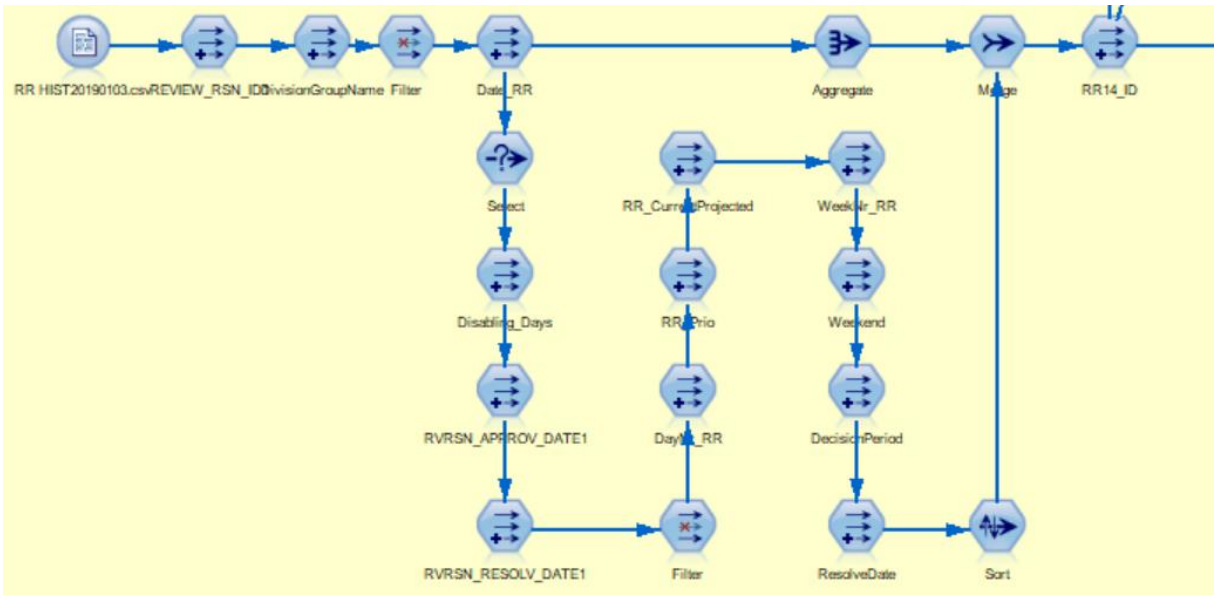

Figure 31: Review reasons

The *adjustments* and *review reasons* are coupled in the next part, which can be seen in Figure 32. In the merge node between the *RR14_ID* and *Select*, the initial decisions on a review reason are coupled. This is done by coupling all *adjustments* and *review reasons* on part number and for which the adjustment date is later than the trigger date, but earlier than the *decision period*. This node splits into two short streams: one gives per exception whether an adjustment has occurred or not, the other examines the adjustments and categorizes the adjustments to *order related, forecast related, setting related* or *other*. In the other *merge* node (coming from the *select* node) adjustments after the *decision period* and before the *resolve date* are examined. These three streams are merged and give per review reason whether the initial decision was to take action or not, what kind of adjustments have been made and whether adjustments have been made after the *decision period*. Next to this information, some general information about the review reason is still kept. The *performance*

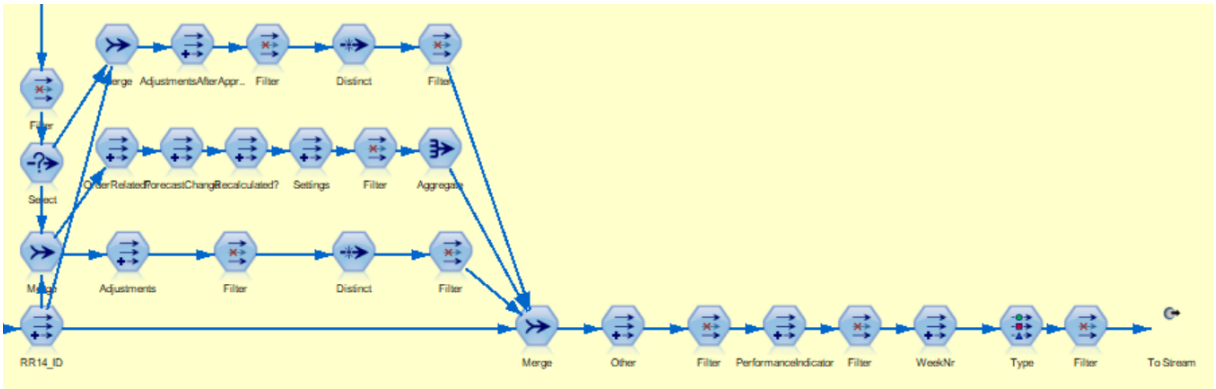

Figure 32: Data set creation

indicator is derived and follows the logic introduced in section 2.3.2. Finally, some columns are renamed and only relevant information is selected in the *filter* node.

Vitality

In this node, which can be seen in Figure 33, the vitality is gathered for all parts from the RTA0 database. This node is probably the simplest node. Since the vitality does hardly change, the data is gathered once and no adjustments are performed. The *filter* is used to only pass the part number and the vitality and to adjust the names of the data.

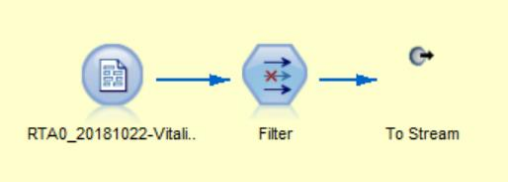

Figure 33: SPSS Model - Vitality

Successor

In this node, which can be seen in Figure 34, the successor information is gathered from the RTA1 table. This gives the information whether a part has a newer part which should be used. This node passes a part number with a yes/no indicator whether this part has a successor.

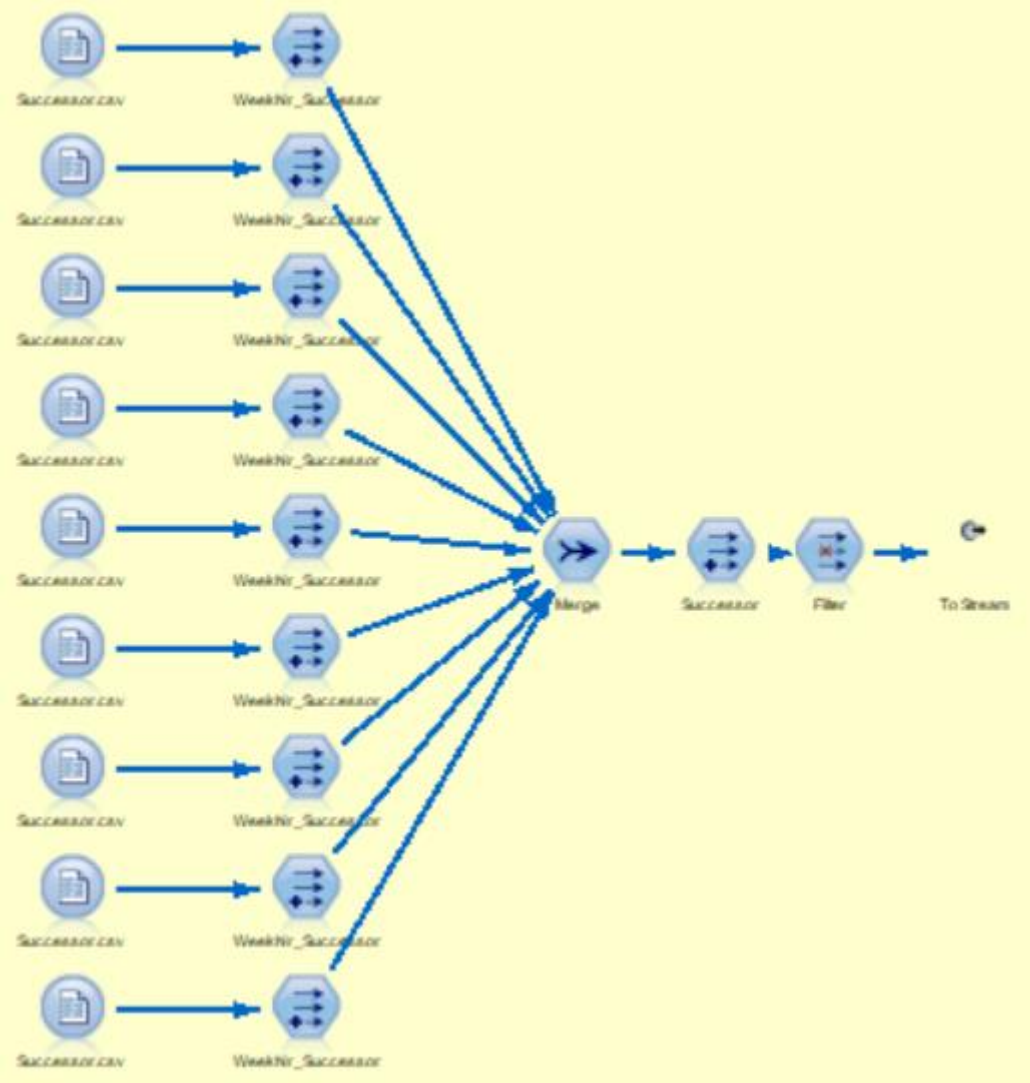

Figure 34: SPSS Model - Successor

Historical Demand

In this node, which can be seen in Figure 35, the historical demand for the last two years and the historical demand for the last four weeks is gathered and calculated from the PTB3 database. Since this node is quite complex, we advise the reader to see the model in SPSS Modeler. In the *select node*, only relevant historical activity (demand) is selected. Next a week number is added. From the week number, two arcs follow: one will calculate the activity in the last four weeks (straight) the other arcs calculate the activity in the last two years (down). The activity is given per period and dependent on the current moment, the activity in the last four weeks should be gathered from the last one or two periods. Since only the activity of the last two years is gathered in the PTB3 database, the activity in the last two years is simply the sum of the activity over a part. However, since review reasons are triggered on the prime part number and planning is done over the sum of demand of all parts related to the prime, the historical demand is summed over all parts connected to the prime.

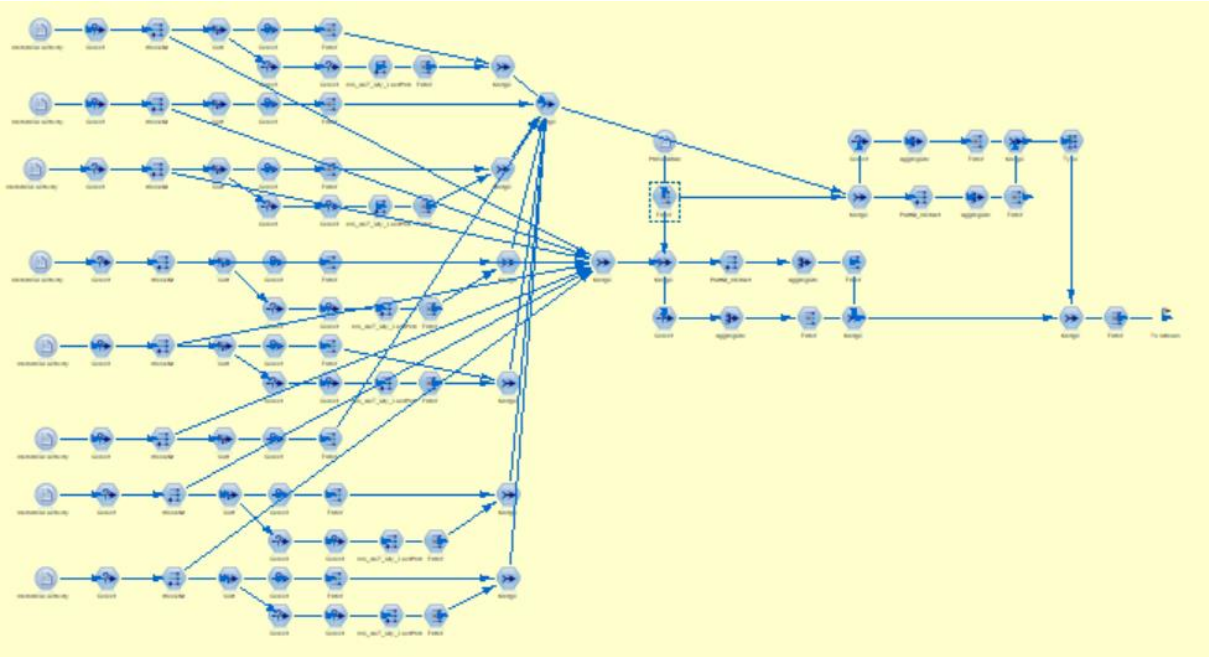

Figure 35: SPSS Model - Historical demand

ReturnRate

In this node, which can be seen in Figure 36, the return rate is calculated based on data gathered from the CPPS database. The number of returned items over the past two years is simply divided by the quantity of historical demand over the past two years.

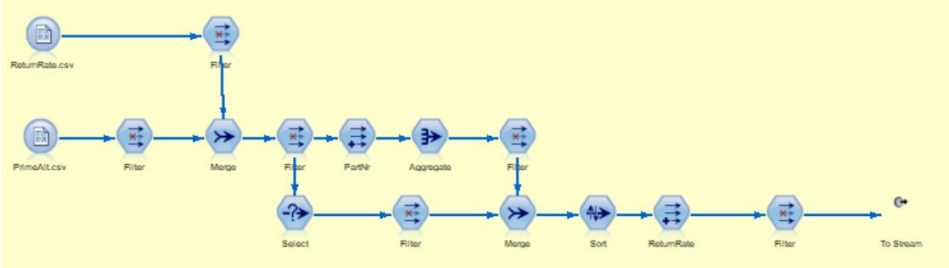

Figure 36: SPSS Model - Return rate

OpenOrders

In this node, which can be seen in Figure 37, information of open orders is gathered form the MTA0 and MTA1 database. The quantity of open orders is calculated for *alternate* and *prime* parts and passed.

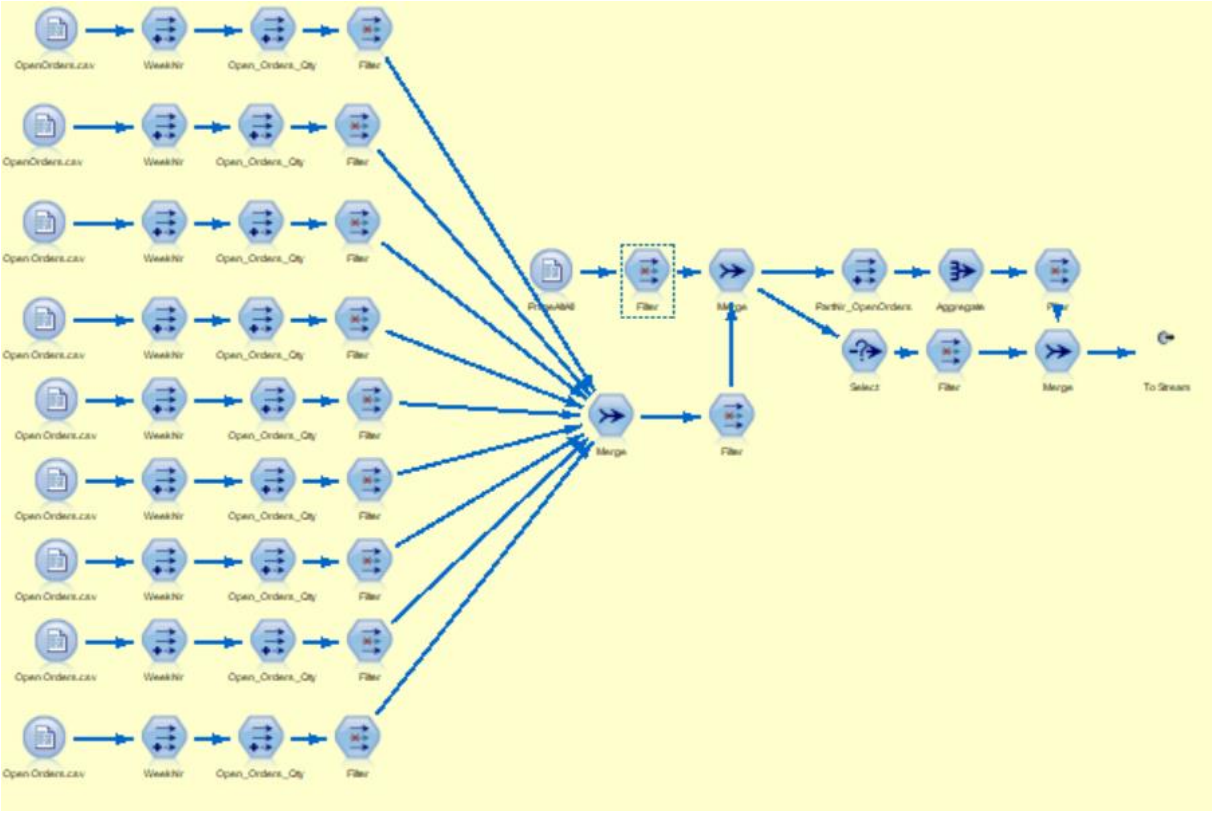

Figure 37: SPSS Model - Open orders

Stock

Every day the current stock position is gathered from the PTB0 database. In this database, different kinds of stock exist: warranty, repair and normal stock. Data is transformed in such a way that for every part, for every day the current warranty, repair and normal stock is passed. This can be seen in Figure 38.

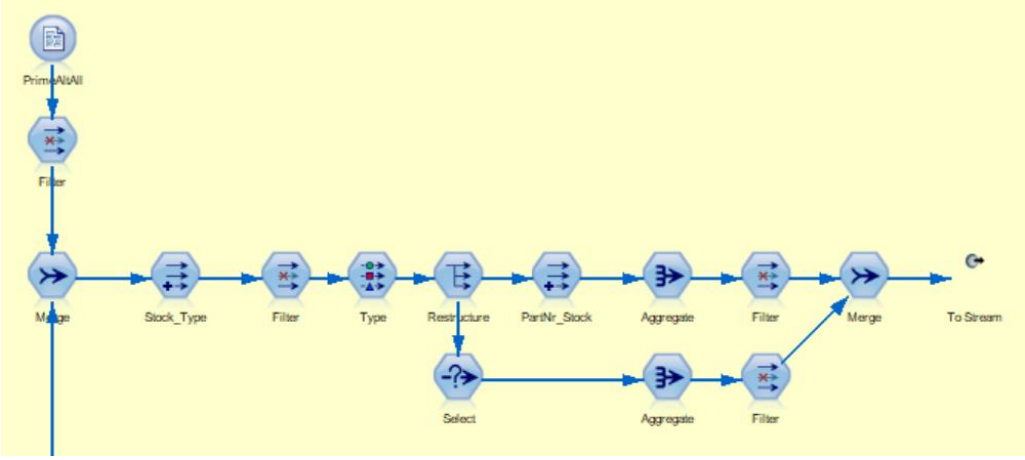

Figure 38: SPSS Model – Stock

CriticalParts

In this node, which can be seen in Figure 39, critical part information is gathered from the PTC4 database. It passes a table with information per part and week number whether a part is critical and by what quantity.

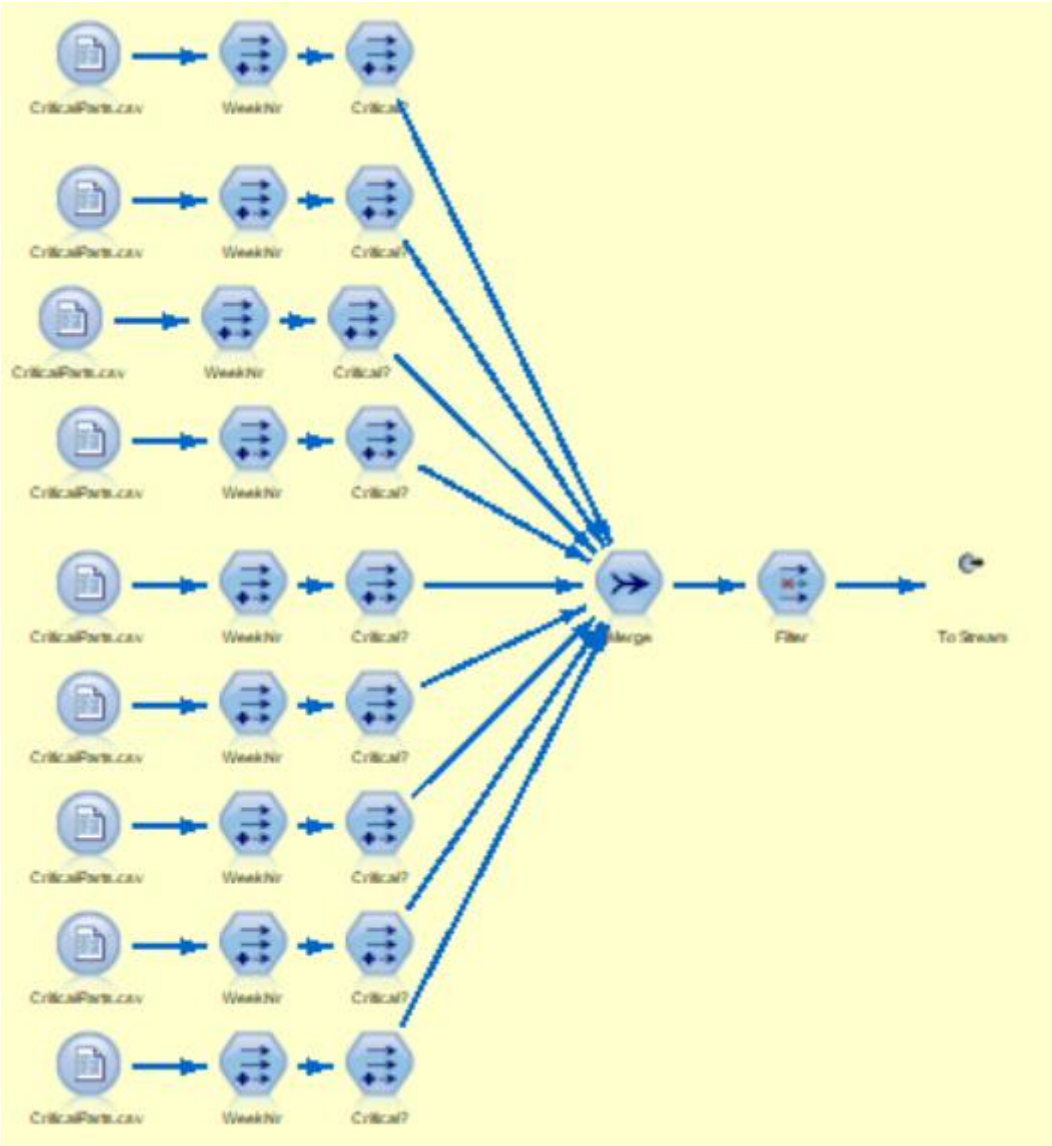

Figure 39: SPSS Model - Critical parts

Shelflife

In this node, which can be seen in Figure 40, the shelf life is gathered for all parts. Since the shelf life will not change for a product, this data is gathered only once (first step). In the second step the shelf life is transformed to years. The *filter* is used to only pass the part number and the shelf life in years.

Figure 40: SPSS Model - Shelf life

PartPlanning

In this node, which can be seen in Figure 41, the part planning levels are gathered from the PTB1 database. Since this should be summed over all parts related to the *prime* parts. This gives per part the critical stock level, the re-order level, the order up to point and the excess level.

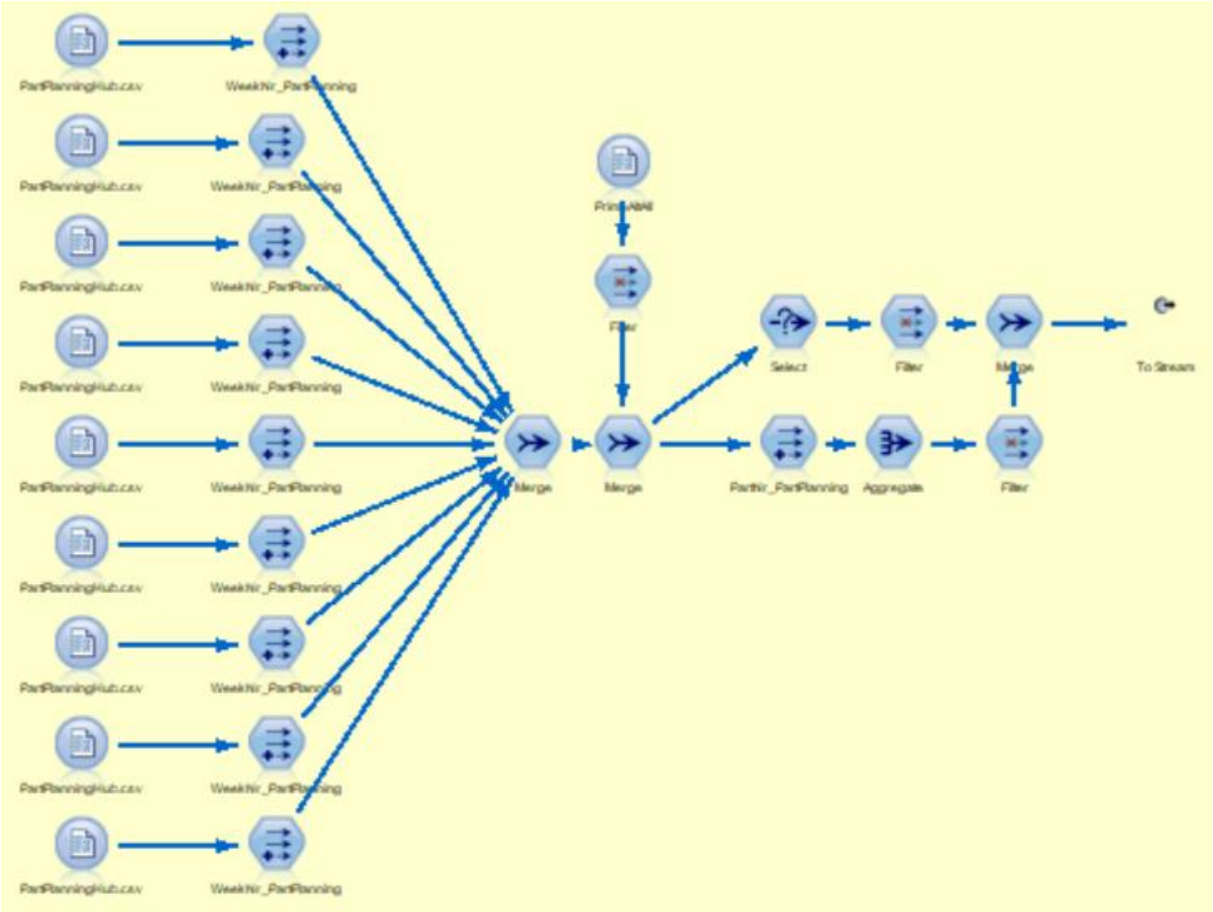

Figure 41: SPSS Model - Part planning

Auto-Order

In this node, which can be seen in Figure 42, auto-order information is gathered from different databases. Only the parts for which one or more order types are disabled for auto-ordering are given. This node passes a yes/no indicator for whether auto-order is disabled for one or more order types.

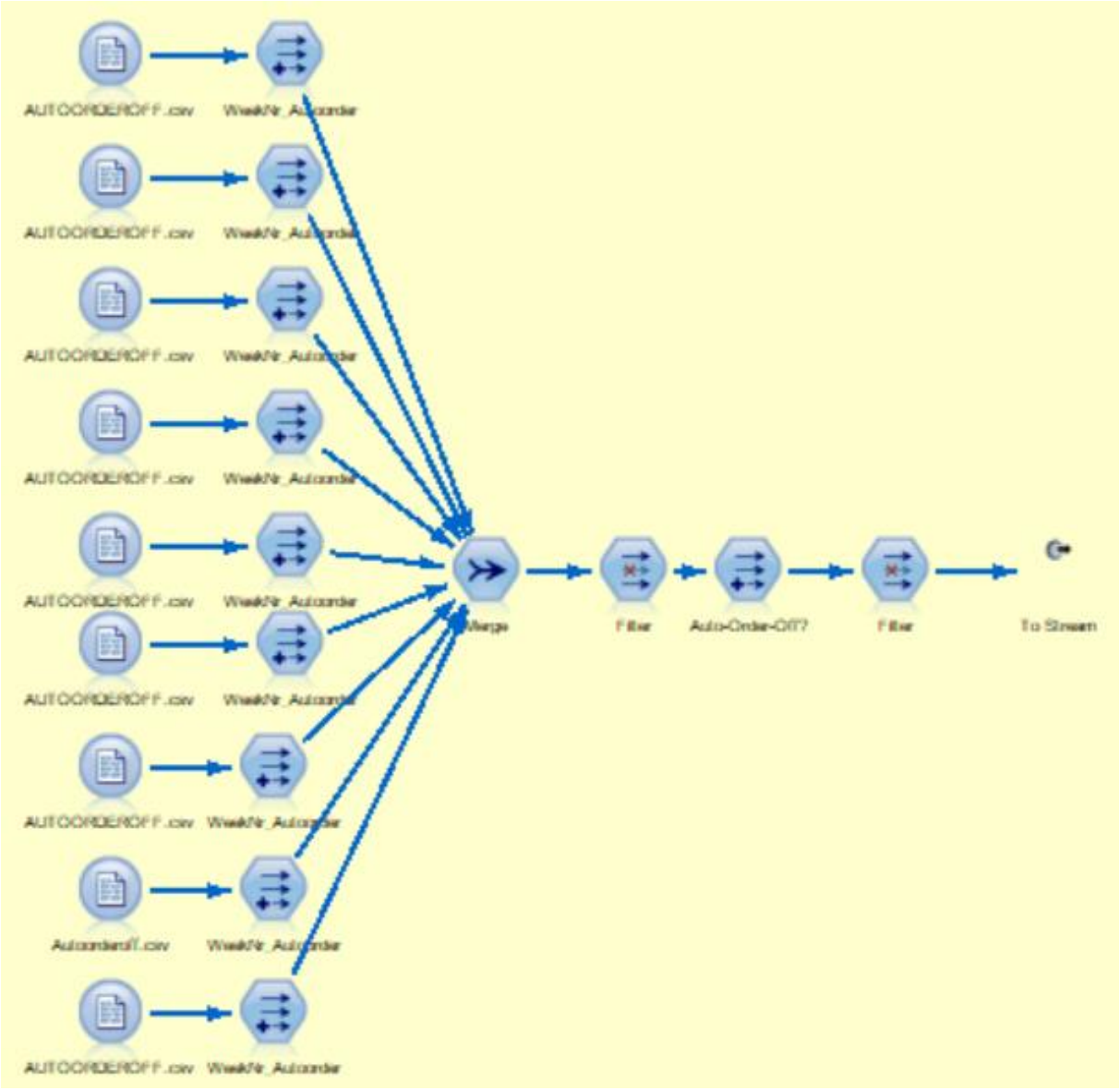

Figure 42: SPSS Model - Auto order

Forecast

In this node, which can be seen in Figure 43, the forecasting method and forecasted demand for the next two years is calculated. The forecasted demand is the sum of all parts related to the *prime* part.

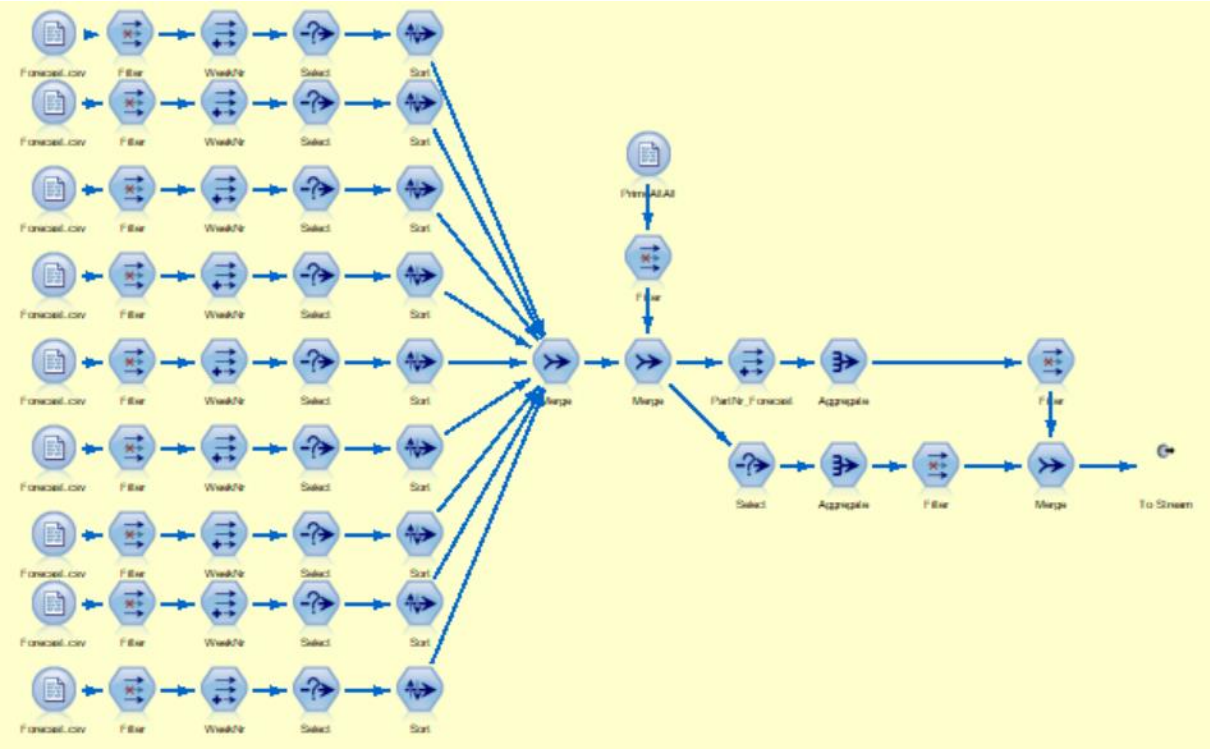

Figure 43: SPSS Model – Forecast

Contracts

In this node, which can be seen in Figure 44, the existing contracts are gathered from the RTC2 database. Every week the contract information is gathered and transformed, such that a table is created which states per part which kind of supplier is currently available to order from, with the given lead time.

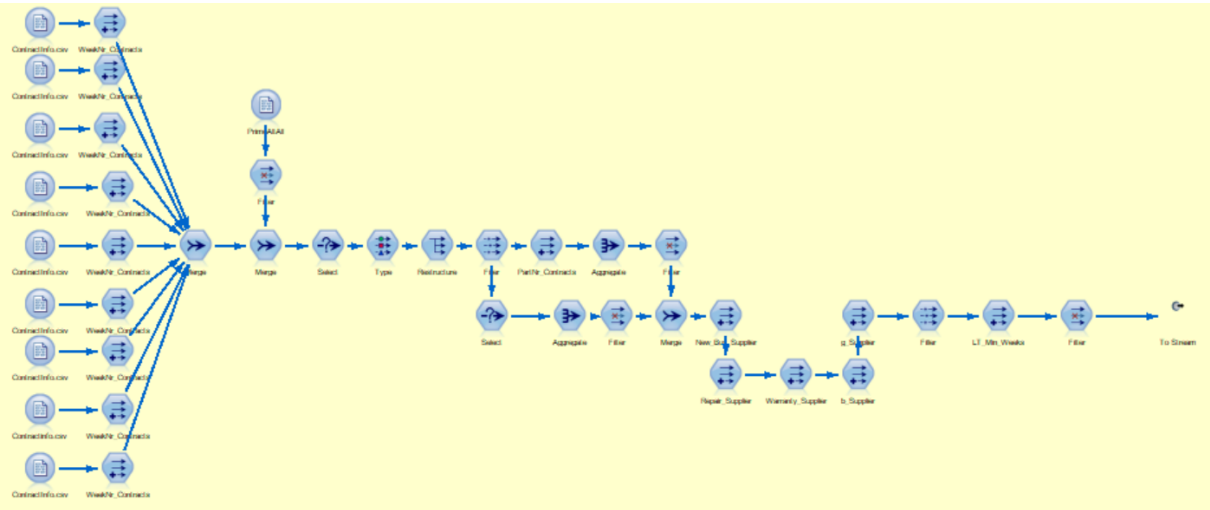

Figure 44: SPSS Model - Contracts

MaterialClass

In this node, which can be seen in Figure 45, the material class is gathered. In the second step, the week number is added to the data and the data is merged. Since the material class consists of two character in which the first one represents the usage x price and the second represents only the usage, the data is split into two separate columns. The *filter* is used to only pass the part number, the week number, the usage x price and the usage.

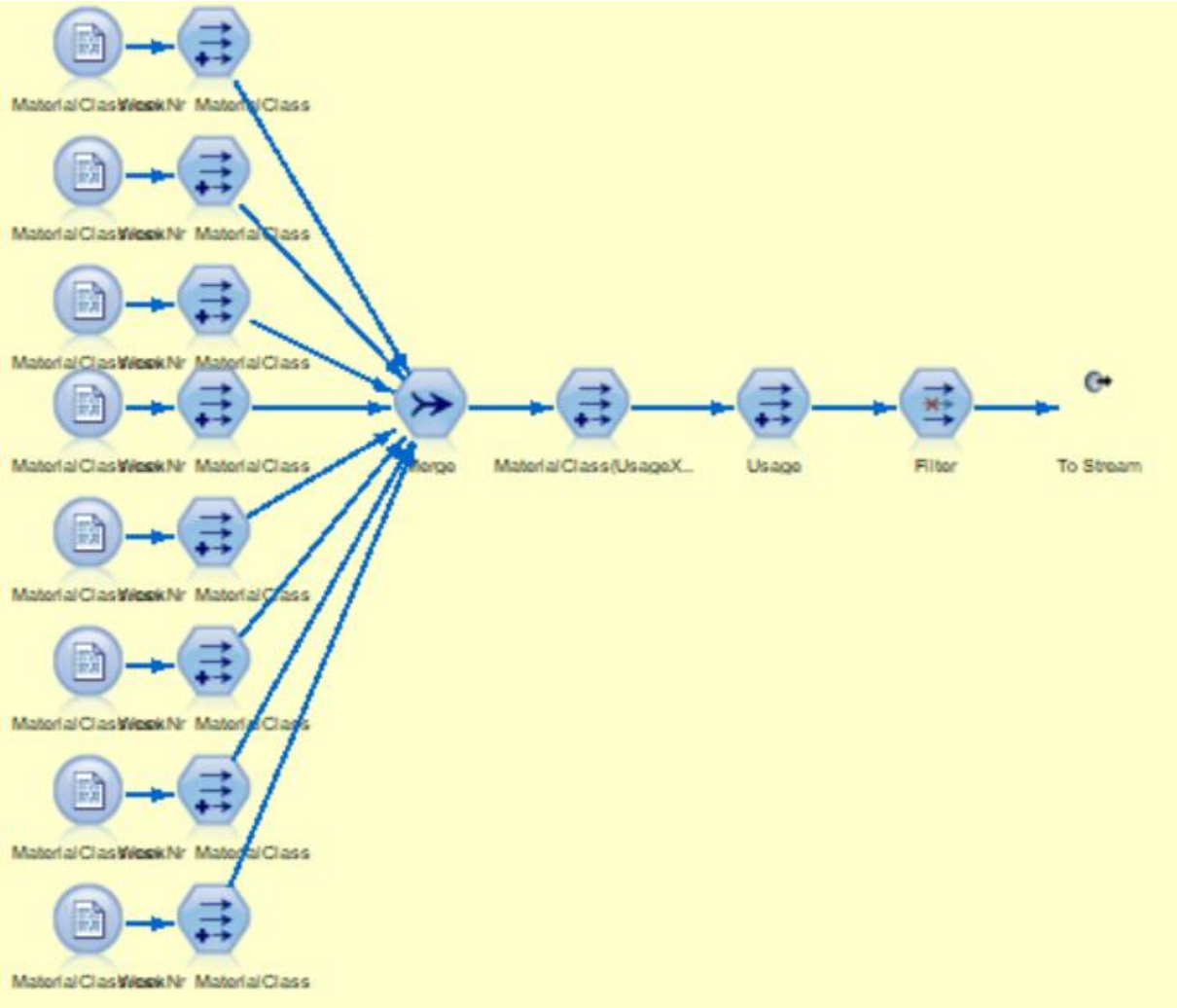

Figure 45: SPSS Model - Material class

PartData

In this node, which can be seen in Figure 46, different useful data is gathered from the RTA6 database. Since some data should be only available from the actual part number and some data a combination of all parts related to the *prime* part, this leads to two sections. Finally, this stream gives the following information per part, per week: birth date, end of service date, first stock date, weighted average cost, air support indicator, analyzer code, c&a code, CI144 indicator, critical part unsatisfied indicator, division owner code, hub responsibility indicator, last time buy indicator, new business opportunity indicator and stocked indicator.

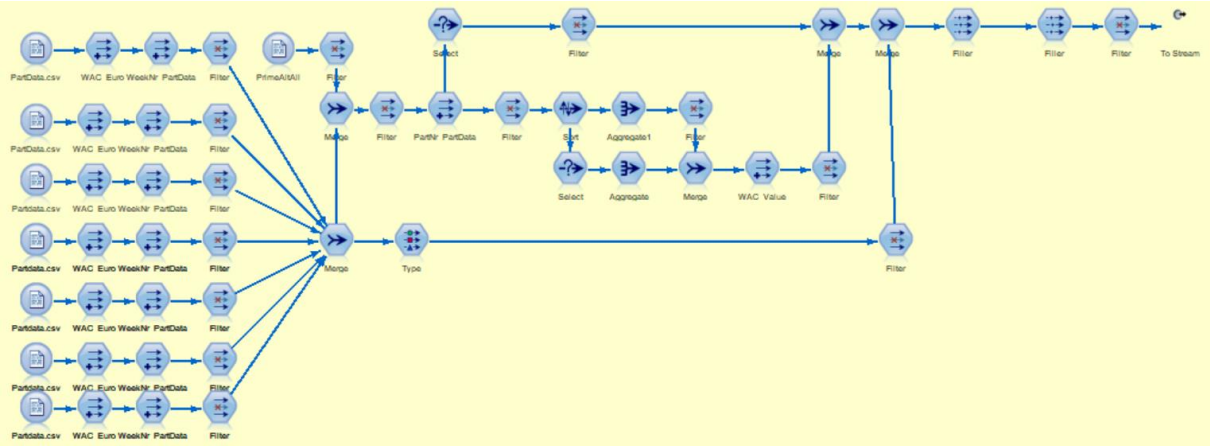

Figure 46: SPSS Model - Part data

Appendix F

This appendix contains the data quality check performed in SPSS Modeler.

Complete fields (%): 67,44% Complete records (%): 0,94%

Field	% Complete
Shelflife	2,279
OpenOrderArrival	37,399
ForecastMethod	65,55
MinimumLeadTime	72,252
ReturnRate	86,193
PSS	86,997
FirstStockDate	89,544
MaterialClass	90,214
Usage	90,214
EndOfServiceDate	91,019
DivisionGroupName	98,928
BirthDate	99,464
C&ACode	99,598
DivisionOwnerCode	99,598
NewBuySupplier	100
RepaierSupplier	100
WarrantySupplier	100
BSupplier	100
GSupplier	100
StockLevel	100
StockLevelRepair	100
StockLevelWarranty	100
CriticalQuantity	100
CriticalIndicator	100
OpenOrderQuantity	100
HistoricalActivity2Years	100
HistoricalActivity1Month	100
Successor	100
Vitality	100
Weekend	100
#OtherRR	100
AutoOrderIndicator	100
ForecastedQuantity	100
WACValue	100
AirSupportIndicator	100
AnalyzerCode	100
CI144MFGXFERIndicator	100
CriticalPartUnsatisfiedIndicator	100
HubResposibleIndicator	100
LasttimeBuyIndicator	100
NBOIndicator	100
NewPartIndicator	100
StockedIndicator	100

Appendix G

This appendix contains different frameworks which compare different algorithms.

Model	Allows $n < p$	Pre-processing	Interpretable	Automatic feature selection	# Tuning parameters	Robust to predictor noise	Computation time
Linear regression†	×	CS, NZV, Corr	✓	×	0	×	✓
Partial least squares	✓	CS	✓	○	1	×	✓
Ridge regression	×	CS, NZV	✓	×	1	×	✓
Elastic net/lasso	×	CS, NZV	✓	✓	1–2	×	✓
Neural networks	✓	CS, NZV, Corr	×	×	2	×	×
Support vector machines	✓	CS	×	×	1–3	×	×
MARS/FDA	✓		○	✓	1–2	○	○
K-nearest neighbors	✓	CS, NZV	×	×	1	○	✓
Single trees	✓		○	✓	1	✓	✓
Model trees/rules†	✓		○	✓	1–2	✓	✓
Bagged trees	✓		×	✓	0	✓	○
Random forest	✓		×	○	0–1	✓	×
Boosted trees	✓		×	✓	3	✓	×
Cubist†	✓		×	○	2	✓	×
Logistic regression*	×	CS, NZV, Corr	✓	×	0	×	✓
{LQRM}DA*	×	NZV	○	×	0–2	×	✓
Nearest shrunken centroids*	✓	NZV	○	✓	1	×	✓
Naïve Bayes*	✓	NZV	×	×	0–1	○	○
C5.0*	✓		○	✓	0–3	✓	×

†regression only *classification only

Symbols represent affirmative (✓), negative (×), and somewhere in between (○)

Figure 47: Framework by Kuhn and Johnson (2013)

	Decision Trees	Neural Networks	Naïve Bayes	kNN	SVM	Rule-learners
Accuracy in general	**	***	*	**	****	**
Speed of learning with respect to number of attributes and the number of instances	***	*	****	****	*	**
Speed of classification	****	****	****	*	****	****
Tolerance to missing values	***	*	****	*	**	**
Tolerance to irrelevant attributes	***	*	**	**	****	**
Tolerance to redundant attributes	**	**	*	**	***	**
Tolerance to highly interdependent attributes (e.g. parity problems)	**	***	*	*	***	**
Dealing with discrete/binary/continuous attributes	****	***(not discrete)	***(not continuous)	***(not directly discrete)	**(not discrete)	***(not directly continuous)
Tolerance to noise	**	**	***	*	**	*
Dealing with danger of overfitting	**	*	***	***	**	**
Attempts for incremental learning	**	***	****	****	**	*
Explanation ability/transparency of knowledge/classifications	****	*	****	**	*	****
Model parameter handling	***	*	****	***	*	***

Figure 48: Framework by Kotsiantis (2007)

Appendix H

This appendix contains a list of the 61 features.

Data	
# of review reasons	*Last Time Buy indicator*
A code	*Lead time indicator*
Air Support Indicator	*Life cycle position*
Analyzer code	*Material Class*
Auto-order indicator	*Minimum lead time*
B supplier	*NBO indicator*
Birth age	*New buy supplier*
Birth date	*New part indicator*
C code	*Open order arrival*
CI144 MFGXFER indicator	*Open order indicator*
Critical indicator	*Open orders quantity*
Critical part unsatisfied indicator	*Planned indicator*
Critical quantity	*Policy safety stock level*
Critical severity	*Repair supplier*
Division group name	*Return rate*
Division owner code	*Return rate indicator*
End of service date	*Runout time*
EOS age	*Shelf life*
EOS Indicator	*Shelf life indicator*
First stock age	*Stock level / inventory level*
First stock date	*Stock repair*
Forecast method	*Stock warranty*
Forecasted quantity	*Stocked indicator*
Fraction inventory position currently available	*Successor*
Fraction recent demand	*Supplier indicator*
G supplier	*Usage*
Historical activity 1 period	*Vitality*
Historical activity 2 years	*WAC*
Hub responsible indicator	*Warranty supplier*
Inventory position	*Weekend*
Inventory position severity	

Appendix I

This appendix contains the approach of the feature selection step and the full results.

We used a *wrapper* method with a bi-directional search. Since WEKA has different algorithms embedded in their software compared to SPSS Modeler, we have chosen to use the J48 decision tree algorithm, which is comparable with the C5.0 decision tree algorithm in SPSS Modeler. The *search termination,* which is the number of consecutive non-improving nodes before terminating the search, is set to 15. The default value is 5, but since our datasets contain a relatively low number of features, we have chosen to increase this value to evaluate more subsets. Next to that, *cross-validation* is used with 5 folds and 1 seed. This means that the algorithm determines 5 times a subset of features which lead to the best results. The settings are shown in Table 28.

Settings	Value
Attribute evaluator	*ClassifierSubsetEval*
Classifier	*J48*
Evaluation measure	*Accuracy*
Use training	*Yes*
Search method	*BestFirst*
Direction	*Bi-directional*
SearchTermination	*15*
Cross-validation	*Folds: 5, Seed: 1*

Table 28: Settings feature selection

Feature selection is performed on the 16 datasets explained in section 4.4. Per feature it is given in how many of the 5 folds the feature was selected. These results are shown in Table 31. Based on these results, Table 32, Table 33, Table 34 and Table 35 are derived. In these tables the number of features which are selected in 0, 1, 2, 3, 4 or 5 of the folds are given. The more often a feature is selected, the more important the feature is for the algorithm. Features which are never selected (0/5) are considered as not relevant, which means they can be excluded from the dataset. Features that appear in all folds (5/5), should be considered as relevant features. For features that are selected sometimes, a consideration should be made. Since decision tree algorithms themselves distinguish relevant and irrelevant features during the modelling stage, we choose to consider features which are selected at least once as a relevant feature. Based on these results, we conclude that a significant number of features is irrelevant, since they are never selected as relevant features. These results are summarized in Table 29. These features will be excluded from the datasets when experiments are performed with relevant features, which is done in chapter 5.

To determine the most relevant features, the average number a feature is selected is calculated, based on all datasets. The most relevant features are shown in Table 30.

	All Imputation	Only good Imputation	All No imputation	Only good No imputation
R24	19	31	22	29
R25	31	40	27	38
R26	30	25	27	26
R83	23	28	21	28

Table 29: Number of irrelevant features

Feature	**Selected**
AnalyzerCode	3.8125
WACValue	3.0625
ReturnRate	2.9375
LifeCyclePosition	2.8125
ForecastedQuantity	2.75
FractionRecentDemand	2.6875
FirstStockAge	2.625
BirthYearAge	2.5625
InventoryPositionSeverity	2.5625
HistoricalActivity2Years	2.5
StockLevel	2.375
Weekend	2.25
HistoricalActivity1Month	2.125
PSS	2.0625
InventoryPosition	2
#RR	1.875
Vitality	1.875
RunOutTime	1.8125
MinimumLeadTime	1.75
OpenOrderQuantity	1.75
CriticalQuantity	1.5
FractionInventoryAvailable	1.5
CriticalSeverity	1.375
EOSAge	1.375
RepairSupplier	1.375
StockLevelRepair	1.125
WarrantySupplier	1.125
AutoOrderIndicator	0.9375
NewBuySupplier	0.9375
CI144MFGXFERIndicator	0.75
OpenOrderIndicator	0.75

Feature	**Selected**
CriticalIndicator	0.6875
EndOfServiceDate	0.625
LeadtimeIndicator	0.625
ReturnRateIndicator	0.625
EOSDateIndicator	0.5625
ShelflifeIndicator	0.5625
DivisionGroupName	0.5
MaterialClass	0.5
OpenOrderArrival	0.5
StockLevelWarranty	0.5
NewPartIndicator	0.4375
SupplierIndicator	0.4375
Usage	0.4375
LasttimeBuyIndicator	0.375
HubResposibleIndicator	0.3125
Shelflife	0.3125
Successor	0.3125
NBOIndicator	0.25
BSupplier	0.1875
CriticalPartUnsatisfiedIndicator	0.1875
DivisionOwnerCode	0.1875
ForecastMethod	0.1875
PlannedIndicator	0.125
ACode	0.0625
CCode	0.0625
AirSupportIndicator	0
BirthDate	0
FirstStockDate	0
GSupplier	0
StockedIndicator	0

Table 30: Feature relevance

Per feature it is shown in how many of the folds the feature was selected. This means that the more a feature is selected, the more important that feature is, whereas features that are not selected are unimportant.

Review reason	R24				R25				R26				R83			
Exceptions	All	Good	All	Good	All	Good	All	Good	All	Good	All	Good	All	Good	All	Good
Imputation yes/no	Imputation		No imputation		Imputation		No imputation		Imputation		No imputation		Imputation		No imputation	
ReturnRate	1	1	2	3	2	2	3	2	2	2	2	2	1	1	2	2
NewBuySupplier	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
RepairSupplier	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
WarrantySupplier	5	5	5	4	5	5	5	5	0	0	1	1	5	5	5	5
BSupplier	2	2	2	2	0	0	1	0	1	2	0	1	2	0	0	0
GSupplier	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MinimumLeadTime	2	3	3	3	3	2	5	2	1	2	0	4	5	1	2	3
Usage	0	0	2	0	0	0	1	0	0	0	0	0	0	0	0	0
StockLevel	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
StockLevelRepair	1	1	2	0	1	2	1	1	2	0	0	0	0	1	0	0
StockLevelWarranty	1	1	2	1	2	0	1	0	0	1	0	0	0	0	2	0
Shelflife	1	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0
PSS	2	1	1	1	1	1	2	1	3	2	3	0	1	1	3	1
CriticalQuantity	3	1	2	1	1	0	2	0	4	1	1	2	1	1	0	2
CriticalIndicator	1	0	0	0	0	0	0	0	0	2	0	3	1	0	1	0
OpenOrderQuantity	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	1
OpenOrderArrival	0	0	0	0	0	1	0	0	5	2	2	0	0	0	0	0
HistoricalActivity2Years	2	3	2	2	2	0	0	0	0	2	1	1	4	0	2	1
HistoricalActivity1Month	0	0	0	0	0	0	3	0	1	1	1	0	1	0	2	0
Successor	3	2	3	3	5	0	4	3	2	0	2	1	4	3	3	4
Vitality	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DivisionGroupName	5	4	4	1	2	2	1	1	3	2	2	2	4	3	4	4
Weekend	0	0	0	0	0	0	0	0	0	0	0	0	1	2	0	0

#RR	1	0	3	0	0	0	1	0	2	2	2	4	1	1	4	3
AutoOrderIndicator	3	2	4	5	2	1	5	1	2	1	3	0	3	3	4	4
ForecastedQuantity	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BirthDate	1	2	2	1	0	2	0	2	2	1	4	2	4	4	4	3
EndOfServiceDate	3	2	5	3	1	3	3	3	2	1	1	1	4	2	4	2
FirstStockDate	0	0	0	0	1	0	0	0	0	0	0	1	2	0	1	0
WACValue	2	2	2	1	1	0	1	2	3	1	2	2	3	2	5	3
AirSupportIndicator	2	2	3	5	2	0	0	1	3	1	1	2	5	5	5	4
AnalyzerCode	0	0	2	0	0	0	0	0	0	1	1	1	0	0	1	0
CI144MFGXFERIndicator	1	0			1	0			1	1			0	1		
CriticalPartUnsatisfiedIndicator	4	4	1	0	5	2	4	0	2	2	2	2	5	4	4	4
HubResposibleIndicator	0	0	0	1	2	0	0	0	0	0	0	1	1	0	2	1
LasttimeBuyIndicator	1	1	1	0	0	1	0	2	3	3	2	4	2	2	3	3
NBOIndicator	1	0	2	0	0	0	0	0	0	0	0	0	0	0	1	0
NewPartIndicator	1	2	0	1	2	0	0	4	1	0	2	0	0	0	2	0
StockedIndicator	2	0	0	0	2	0	0	0	0	1	0	1	0	0	1	0
ShelflifeIndicator	1	0	0	0	0	0	2	0	0	2	2	1	0	0	0	0
LeadtimeIndicator	2	0	1	1	1	0	1	0	0	1	0	0	0	0	3	2
ReturnRateIndicator	1	2	2	2	1	3	2	2	1	2	2	2	2	0	1	3
PlannedIndicator	0	0			1	0			0	0			0	0		
EOSDateIndicator	2	0	4	0	2	1	1	1	3	2	2	3	1	3	4	4
CCode	1	1	1	2	3	0	2	0	0	1	2	1	2	2	2	2
ACode	2	1	4	3	2	4	2	0	5	3	3	2	3	5	5	3
FirstStockAge	0	0			0	1			1	0			0	3		
BirthYearAge	2	1	4	1	0	0	0	0	2	3	3	2	4	2	4	1
EOSAge	1	0	0	0	0	0	0	0	0	0	0	0	1	3	0	0
LifeCyclePosition	1	1	1	2	0	0	1	0	0	0	0	0	1	1	0	1
SupplierIndicator	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
OpenOrderIndicator	5	3	5	2	0	1	2	0	0	2	3	1	5	0	4	5
InventoryPosition	3	1	1	1	0	0	1	0	2	0	2	0	1	3	1	2

UNIVERSITY OF TWENTE.

FractionRecentDemand	0	2	0	0	0	0	0	0	2	0	0	0	2	0	2	0
FractionInventoryAvailable	0	0	1	1	0	0	1	0	0	0	0	0	1	1	0	0
CriticalSeverity	1	0	0	0	0	0	1	1	0	0	0	1	0	1	2	0
RunOutTime	1	1	0	0	0	0	0	0	0	2	0	0	1	2	0	0
InventoryPositionSeverity	3	0	2	1	2	2	3	3	2	1	1	2	3	0	2	3
MaterialClass	3	3	3	2	4	2	4	3	5	1	3	1	5	1	5	4
DivisionOwnerCode	1	1	1	0	0	3	0	1	1	1	1	2	2	2	2	0
ForecastMethod	3	0	4	1	2	2	3	2	4	1	3	3	3	1	3	1

Table 31: Results feature selection

R24	All Imputation	Only good Imputation	All No imputation	Only good No imputation
Selected in 0/5	19	31	21	29
Selected in 1/5	19	13	10	14
Selected in 2/5	11	10	13	7
Selected in 3/5	8	4	5	5
Selected in 4/5	1	2	6	1
Selected in 5/5	3	1	3	2

Table 32: Results feature selection (R24)

R25	All Imputation	Only good Imputation	All No imputation	Only good No imputation
Selected in 0/5	31	40	27	37
Selected in 1/5	11	7	13	8
Selected in 2/5	13	9	7	7
Selected in 3/5	2	3	5	4
Selected in 4/5	1	1	3	1
Selected in 5/5	3	1	3	1

Table 33: Results feature selection (R25)

R26	All Imputation	Only good Imputation	All No imputation	Only good No imputation
Selected in 0/5	30	25	26	26
Selected in 1/5	8	18	9	14
Selected in 2/5	12	15	15	12
Selected in 3/5	6	3	7	3
Selected in 4/5	2	0	1	3
Selected in 5/5	3	0	0	0

Table 34: Results feature selection (R26)

R83	All Imputation	Only good Imputation	All No imputation	Only good No imputation
Selected in 0/5	23	28	20	28
Selected in 1/5	14	13	7	7
Selected in 2/5	7	8	12	6
Selected in 3/5	5	7	5	8
Selected in 4/5	6	2	9	7
Selected in 5/5	6	3	5	2

Table 35: Results feature selection (R83)

Appendix J

This appendix contains the results of the *auto-classifier* used in SPSS Modeler on the different datasets. Per dataset, the top three algorithms are given with their *accuracy*.

R24	All Imputation		Only good Imputation		All No imputation		Only good No imputation	
Feature selection	Tree-As XGBoost CHAID	57.6 56.1 55.4	Random trees Tree-As QUEST	61.3 60.0 60.0	Random trees C5 XGBoost	60.7 60.0 57.9	XGBoost Chaid Random trees	67.5 63.8 58.8
No Feature selection	Tree-As C5 Random trees	57.6 56.1 56.1	Random trees Chaid XGBoost	67.5 65.0 63.8	C5 Random trees XGBoost	60.7 59.3 57.1	Chaid C5 Random trees	68.8 63.8 63.8

R25	All Imputation		Only good Imputation		All No imputation		Only good No imputation	
Feature selection	XGBoost C5.0 Tree-As	72.2 67.8 67.0	C5 Tree-As XGBoost	75.6 69.5 66.1	C5 Tree-As C&R	76.5 73.9 65.2	C5 XGBoost C&R	74.6 64.4 64.4
No Feature selection	Random trees Tree-As QUEST	68.7 67.8 67.0	Random trees C5 QUEST	76.3 74.6 72.9	C5 Tree-As C&R	76.5 73.9 66.1	Chaid XGBoost QUEST	64.4 62.7 62.7

R26	All Imputation		Only good Imputation		All No imputation		Only good No imputation	
Feature selection	Tree-As Chaid C5.0	63.9 59.7 58.3	XGBoost C5 Tree-As	62.9 61.4 61.4	Tree-As Chaid QUEST	57.1 55.4 53.6	Tree-As C&R XGBoost	66.2 63.4 60.6
No Feature selection	XGBoost C&R C5	63.4 58.0 55.4	C5 Tree-As C&R	62.9 61.4 61.4	Random trees C&R XGBoost	56.3 56.3 55.3	C&R C5 Quest	70.4 62.0 62.0

R83	All Imputation		Only good Imputation		All No imputation		Only good No imputation	
Feature selection	Tree-As Random trees XGBoost	63.3 62.8 62.8	XGBoost C5 Random trees	69.8 68.3 65.3	Random trees Tree-As XGBoost	64.7 64.1 63	C&R Chaid Tree-As	67.0 66.0 65.5
No Feature selection	C5 XGBoost Tree-As	63.3 62.5 58.9	C5 XGBoost C&R	69.3 68.8 66.8	C5 Chaid QUEST	63.8 63.6 62.6	Tree-As Quest XGBoost	67.0 65.0 64.0

Appendix K

This appendix contains the full results of the models made by the algorithms with *cross validation* and the performance of the ensemble models. The experiments are as explained in section 5.1 and the algorithms settings are explained in section 5.3. The first number in the *algorithm column* is the *minimum number of records in the child node* and the second number is the *pruning severity*. Furthermore, *Acc* is the *accuracy, AUC* is the *Area Under the Curve* and *Prec* is the *precision*. The colors give an indication of the performance compared to other experiments and settings within the review reason. A green color indicates a relatively high performance, whereas a red color indicates a relatively poor performance. A **** in the table indicates that SPSS Modeler was not able to evaluate the results of an ensemble model.

Experiment	Algorithm	R24			R25			R26			R83		
	settings	Acc	AUC	Prec	Acc	AUC	Prec	Acc	AUC	Prec	Acc	AUC	Prec
1	2 - 75	0.602	0.603	0.593	0.692	0.669	0.718	0.552	0.562	0.608	0.623	0.641	0.622
	5 - 75	0.59	0.581	0.586	0.676	0.647	0.703	0.518	0.531	0.582	0.6	0.629	0.605
	2 - 25	0.602	0.603	0.593	0.692	0.674	0.718	0.538	0.547	0.61	0.621	0.636	0.604
	5 - 25	0.592	0.583	0.587	0.678	0.632	0.709	0.518	0.531	0.582	0.602	0.631	0.608
2	2 - 75	0.61	0.612	0.608	0.689	0.65	0.711	0.507	0.513	0.53	0.621	0.661	0.624
	5 - 75	0.6	0.614	0.595	0.689	0.639	0.703	0.514	0.516	0.547	0.609	0.641	0.611
	2 - 25	0.612	0.613	0.61	0.689	0.65	0.711	0.504	0.519	0.529	0.624	0.663	0.626
	5 - 25	0.607	0.617	0.603	0.686	0.631	0.7	0.483	0.483	0.515	0.607	0.645	0.609
3	2 - 75	0.616	0.644	0.598	0.673	0.64	0.701	0.566	0.565	0.571	0.616	0.649	0.628
	5 - 75	0.584	0.616	0.56	0.665	0.637	0.7	0.554	0.55	0.567	0.621	0.657	0.635
	2 - 25	0.621	0.658	0.602	0.673	0.64	0.701	0.559	0.57	0.586	0.616	0.651	0.629
	5 - 25	0.606	0.632	0.596	0.665	0.637	0.7	0.536	0.544	0.553	0.62	0.659	0.631
4	2 - 75	0.615	0.615	0.601	0.677	0.626	0.7	0.557	0.536	0.566	0.633	0.662	0.64
	5 - 75	0.619	0.632	0.61	0.669	0.646	0.699	0.553	0.54	0.565	0.623	0.652	0.634
	2 - 25	0.619	0.626	0.602	0.677	0.626	0.7	0.553	0.534	0.561	0.633	0.668	0.639
	5 - 25	0.616	0.634	0.602	0.666	0.632	0.693	0.553	0.54	0.565	0.622	0.65	0.634
5	2 - 75	0.635	0.674	0.639	0.744	0.747	0.753	0.469	0.45	0.555	0.667	0.659	0.68
	5 - 75	0.635	0.616	0.642	0.703	0.656	0.721	0.501	0.493	0.57	0.656	0.649	0.68
	2 - 25	0.635	0.674	0.639	0.744	0.747	0.753	0.49	0.461	0.568	0.66	0.658	0.682
	5 - 25	0.635	0.616	0.642	0.744	0.716	0.754	0.527	0.487	0.599	0.651	0.642	0.68
6	2 - 75	0.621	0.643	0.635	0.68	0.576	0.678	0.566	0.495	0.589	0.673	0.665	0.675
	5 - 75	0.635	0.616	0.642	0.685	0.595	0.682	0.57	0.561	0.611	0.647	0.637	0.664
	2 - 25	0.635	0.665	0.641	0.737	0.663	0.754	0.522	0.526	0.593	0.665	0.663	0.673
	5 - 25	0.635	0.616	0.642	0.719	0.698	0.729	0.544	0.534	0.616	0.639	0.64	0.664
7	2 - 75	0.626	0.618	0.632	0.678	0.674	0.723	0.628	0.5	0.628	0.659	0.661	0.669
	5 - 75	0.596	0.591	0.621	0.722	0.694	0.739	0.628	0.5	0.628	0.657	0.643	0.67
	2 - 25	0.634	0.619	0.633	0.694	0.705	0.745	0.604	0.555	0.643	0.666	0.679	0.68
	5 - 25	0.627	0.625	0.629	0.727	0.701	0.741	0.585	0.526	0.63	0.668	0.665	0.679
8	2 - 75	0.646	0.634	0.645	0.717	0.702	0.738	0.609	0.53	0.646	0.667	0.655	0.669
	5 - 75	0.593	0.572	0.611	0.722	0.694	0.739	0.601	0.541	0.649	0.667	0.643	0.67
	2 - 25	0.643	0.622	0.644	0.706	0.694	0.746	0.6	0.544	0.662	0.669	0.666	0.673
	5 - 25	0.614	0.593	0.625	0.727	0.701	0.741	0.589	0.55	0.647	0.668	0.656	0.676

Table 36: Results cross validation

Experiment	Algorithm	R24			R25			R26			R83		
	settings	Acc	AUC	Prec	Acc	AUC	Prec	Acc	AUC	Prec	Acc	AUC	Prec
1	2 - 75	0.624	0.631	0.565	0.677	0.65	0.672	0.604	0.602	0.68	0.654	0.654	0.688
	5 - 75	0.648	0.651	0.594	0.602	0.556	0.603	0.637	0.635	0.75	0.648	0.65	0.691
	2 - 25	0.624	0.631	0.565	0.677	0.65	0.672	0.593	0.591	0.667	0.648	0.647	0.676
	5 - 25	0.64	0.644	0.585	0.602	0.556	0.603	0.637	0.635	0.75	0.639	0.641	0.685
2	2 - 75	0.632	0.63	0.586	0.645	0.606	0.635	0.505	0.506	0.5	0.63	0.631	0.671
	5 - 75	0.632	0.635	0.578	0.634	0.593	0.627	0.538	0.539	0.532	0.645	0.647	0.687
	2 - 25	0.632	0.63	0.586	0.645	0.606	0.635	0.495	0.494	0.488	0.627	0.629	0.669
	5 - 25	0.632	0.636	0.576	0.634	0.593	0.627	0.527	0.527	0.523	0.636	0.638	0.681
3	2 - 75	0.528	0.524	0.574	0.639	0.594	0.712	0.505	0.505	0.532	0.596	0.597	0.631
	5 - 75	0.544	0.545	0.597	0.691	0.64	0.739	0.516	0.62	0.536	0.619	0.62	0.655
	2 - 25	0.528	0.524	0.574	0.649	0.602	0.716	0.527	0.531	0.564	0.596	0.597	0.631
	5 - 25	0.544	0.545	0.597	0.691	0.64	0.739	0.527	0.522	0.545	0.741	0.744	0.789
4	2 - 75	0.536	0.536	0.587	0.67	0.617	0.725	0.516	0.515	0.542	0.608	0.607	0.638
	5 - 75	0.544	0.545	0.597	0.701	0.648	0.743	0.516	0.515	0.542	0.625	0.627	0.665
	2 - 25	0.536	0.536	0.587	0.67	0.617	0.725	0.538	0.536	0.56	0.605	0.604	0.634
	5 - 25	0.544	0.545	0.597	0.691	0.633	0.732	0.516	0.515	0.542	0.617	0.619	0.66
5	2 - 75	0.644	0.639	0.59	0.733	0.75	0.643	0.571	0.493	0.692	0.66	0.617	0.646
	5 - 75	0.644	0.64	0.596	0.733	0.75	0.643	0.714	0.563	0.725	0.695	0.66	0.678
	2 - 25	0.644	0.639	0.59	0.733	0.75	0.643	0.625	0.548	0.725	0.67	0.636	0.664
	5 - 25	0.644	0.64	0.596	0.733	0.75	0.643	0.661	0.557	0.727	0.72	0.71	0.711
6	2 - 75	0.644	0.639	0.59	****	****	****	0.696	0.5	0.696	0.64	0.593	0.63
	5 - 75	0.644	0.64	0.596	****	0.646	****	0.625	0.465	0.68	0.685	0.646	0.667
	2 - 25	0.616	0.611	0.571	0.733	0.627	0.633	0.679	0.587	0.744	0.665	0.627	0.656
	5 - 25	0.644	0.64	0.596	0.733	0.755	0.633	0.589	0.506	0.7	0.685	0.651	0.673
7	2 - 75	0.658	0.664	0.605	0.667	0.601	0.75	0.571	0.5	0.571	0.644	0.617	0.65
	5 - 75	0.699	0.709	0.63	0.667	0.601	0.75	0.571	0.5	0.571	0.634	0.61	0.647
	2 - 25	0.658	0.664	0.605	0.667	0.601	0.75	0.536	0.516	0.583	0.644	0.62	0.655
	5 - 25	0.685	0.698	0.612	0.667	0.601	0.75	0.589	0.557	0.61	0.614	0.588	0.631
8	2 - 75	0.644	0.65	0.595	0.667	0.601	0.75	0.571	0.5	0.571	0.639	0.611	0.646
	5 - 75	0.685	0.693	0.628	0.667	0.601	0.75	0.589	0.531	0.588	0.644	0.617	0.65
	2 - 25	0.644	0.65	0.595	0.667	0.601	0.75	0.589	0.557	0.61	0.649	0.621	0.653
	5 - 25	0.685	0.694	0.622	0.667	0.601	0.75	0.607	0.568	0.614	0.644	0.617	0.65

Table 37: Results ensemble models

Appendix L

This appendix contains the decision trees created during cross validation for review reason R25 and R83.

R25

Fold 1:

AnalyzerCode in ["001" "04B" "071" "07B" "08F" "08Y" "09P" "09R" "09X" "0M7"] [Mode: Yes] => Yes
AnalyzerCode in ["005"] [Mode: No]
    Vitality in [1] [Mode: No]
        #RR in [1] [Mode: No] => No
        #RR in [2 3] [Mode: Yes] => Yes
    Vitality in [2 3 5] [Mode: Yes] => Yes
AnalyzerCode in ["008" "014" "01B" "01O" "021" "024" "030" "03E" "03V" "05C" "085" "08C" "08L" "08X" "08Z" "09M" "0C6" "0G3" "0G7" "0V7" "0W7"] [Mode: No] => No
AnalyzerCode in ["018" "020"] [Mode: No] => No
AnalyzerCode in ["01A"] [Mode: No]
    Vitality in [1] [Mode: No]
        WACValue <= 432,010 [Mode: Yes] => Yes
        WACValue > 432,010 [Mode: No] => No
    Vitality in [2 3 5] [Mode: No] => No

Fold 2:

AnalyzerCode in ["001"] [Mode: Yes]
    LifeCyclePosition <= 0,762 [Mode: Yes] => Yes
    LifeCyclePosition > 0,762 [Mode: No] => No
AnalyzerCode in ["005"] [Mode: No]
    HistoricalActivity1Month <= 0,750 [Mode: Yes]
        Vitality in [1] [Mode: No]
            #RR in [1] [Mode: No] => No
            #RR in [2 3] [Mode: Yes] => Yes
        Vitality in [2 3 5] [Mode: Yes] => Yes
    HistoricalActivity1Month > 0,750 [Mode: No] => No
AnalyzerCode in ["008" "01O" "021" "024" "030" "03E" "03V" "085" "08C" "08L" "08X" "09M" "0C6" "0G3" "0G7" "0V7" "0W7"] [Mode: No] => No
AnalyzerCode in ["014"] [Mode: No]
    BirthYearAge <= 4,500 [Mode: Yes] => Yes
    BirthYearAge > 4,500 [Mode: No] => No
AnalyzerCode in ["018" "020" "04B" "071" "07B" "09P" "09R" "09X" "0M7"] [Mode: Yes] => Yes
AnalyzerCode in ["01A"] [Mode: No]
    #RR in [1] [Mode: No] => No
    #RR in [2 3] [Mode: Yes] => Yes
AnalyzerCode in ["01B" "08F" "08Y"] [Mode: No] => No
AnalyzerCode in ["05C"] [Mode: No]
    ReturnRate <= 0 [Mode: Yes] => Yes
    ReturnRate > 0 [Mode: No] => No
AnalyzerCode in ["08Z"] [Mode: No]
    ReturnRate <= 0,458 [Mode: No] => No
    ReturnRate > 0,458 [Mode: Yes] => Yes

Fold 3:

AnalyzerCode in ["001"] [Mode: Yes]
    CI144MFGXFERIndicator = Y [Mode: Yes]
        PSS <= 17 [Mode: No] => No
        PSS > 17 [Mode: Yes] => Yes
    CI144MFGXFERIndicator = N [Mode: Yes] => Yes
AnalyzerCode in ["005" "008" "01A" "01B" "024" "030" "03E" "03V" "085" "08L" "08X" "08Z" "09M" "0C6" "0G7" "0W7"] [Mode: No] => No
AnalyzerCode in ["014" "018" "020" "071" "07B" "08F" "08Y" "09P" "09R" "09X" "0M7"] [Mode: Yes] => Yes
AnalyzerCode in ["01O" "04B" "08C" "0V7"] [Mode: No] => No
AnalyzerCode in ["021"] [Mode: No]
    PSS <= 12,500 [Mode: No] => No
    PSS > 12,500 [Mode: Yes] => Yes
AnalyzerCode in ["05C"] [Mode: No]
    ReturnRate <= 0 [Mode: Yes] => Yes
    ReturnRate > 0 [Mode: No] => No
AnalyzerCode in ["0G3"] [Mode: No]
    ForecastedQuantity <= 6,050 [Mode: No] => No
    ForecastedQuantity > 6,050 [Mode: Yes] => Yes

Fold 4:
AnalyzerCode in ["001" "018" "020" "04B" "07B" "08F" "08Y" "09P" "09R" "09X" "0M7"] [Mode: Yes] => Yes
AnalyzerCode in ["005"] [Mode: No]
    #RR in [1] [Mode: No]
        Vitality in [1] [Mode: No] => No
        Vitality in [2 3 5] [Mode: Yes] => Yes
    #RR in [2 3] [Mode: Yes] => Yes
AnalyzerCode in ["008" "01B" "01O" "021" "024" "030" "03E" "03V" "05C" "08C" "08L" "08Z" "09M" "0C6" "0G7" "0V7" "0W7"] [Mode: No] => No
AnalyzerCode in ["014"] [Mode: Yes]
    BirthYearAge <= 4,500 [Mode: Yes] => Yes
    BirthYearAge > 4,500 [Mode: No] => No
AnalyzerCode in ["01A"] [Mode: No]
    #RR in [1] [Mode: No]
        ForecastedQuantity <= 28,600 [Mode: Yes]
            WACValue <= 448,795 [Mode: Yes] => Yes
            WACValue > 448,795 [Mode: No] => No
        ForecastedQuantity > 28,600 [Mode: No] => No
    #RR in [2 3] [Mode: Yes] => Yes
AnalyzerCode in ["071" "085" "08X"] [Mode: No] => No
AnalyzerCode in ["0G3"] [Mode: No]
    ForecastedQuantity <= 6,050 [Mode: No] => No
    ForecastedQuantity > 6,050 [Mode: Yes] => Yes

Fold 5:
AnalyzerCode in ["001" "008" "01B" "01O" "021" "024" "030" "03E" "03V" "05C" "085" "08C" "08X" "08Z" "09M" "0G7" "0V7" "0W7"] [Mode: No] => No
AnalyzerCode in ["005"] [Mode: No]
    CI144MFGXFERIndicator = Y [Mode: Yes] => Yes
    CI144MFGXFERIndicator = N [Mode: No]
        WACValue <= 83,300 [Mode: Yes]
            WACValue <= 11,135 [Mode: No] => No
            WACValue > 11,135 [Mode: Yes] => Yes
        WACValue > 83,300 [Mode: No] => No
AnalyzerCode in ["014" "018" "020" "04B" "071" "07B" "08F" "08Y" "09P" "09R" "09X"] [Mode: Yes] => Yes
AnalyzerCode in ["01A"] [Mode: No]
    #RR in [1] [Mode: No] => No
    #RR in [2 3] [Mode: Yes] => Yes
AnalyzerCode in ["08L" "0C6" "0M7"] [Mode: No] => No
AnalyzerCode in ["0G3"] [Mode: No]
    ForecastedQuantity <= 3,450 [Mode: No] => No
    ForecastedQuantity > 3,450 [Mode: Yes] => Yes

R83

Fold 1:
AnalyzerCode in ["001" "004" "008" "01A" "01B" "01X" "01Z" "026" "027" "03N" "04B" "04F" "05C" "08Z" "099" "09J" "09M" "0C6" "0I3" "0U7"] [Mode: No] => No
AnalyzerCode in ["005"] [Mode: No]
    FractionInventoryAvailable <= 0,197 [Mode: Yes] => Yes
    FractionInventoryAvailable > 0,197 [Mode: No]
        Weekend = Weekend [Mode: No]
            BirthYearAge <= 3,500 [Mode: Yes] => Yes
            BirthYearAge > 3,500 [Mode: No] => No
        Weekend = Week [Mode: No] => No
AnalyzerCode in ["012" "023" "030" "03D" "03E" "04J" "085" "08C" "08F" "09E" "09R" "09X" "09Y" "0A7" "0M7" "0V7" "0W9"] [Mode: Yes] => Yes
AnalyzerCode in ["013" "018" "0V9"] [Mode: No] => No
AnalyzerCode in ["014"] [Mode: No]
    PSS <= 27,450 [Mode: Yes]
        FirstStockAge in [0 1 2 3 4] [Mode: Yes] => Yes
        FirstStockAge in [5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 22 23 53] [Mode: No]
            HistoricalActivity1Month <= 1,125 [Mode: No] => No
            HistoricalActivity1Month > 1,125 [Mode: Yes] => Yes
    PSS > 27,450 [Mode: No] => No
AnalyzerCode in ["021"] [Mode: No]
    ShelflifeIndicator = Yes [Mode: Yes] => Yes
    ShelflifeIndicator = No [Mode: No]
        DivisionOwnerCode in ["13" "26" "2D" "44" "48" "4S" "71" "7G" "7H" "APL" "ASC" "BOI" "DUN" "G9" "LP" "LQ" "LU" "LX" "MB" "MC" "MN" "MP" "MR" "MS" "MT" "MV" "OBB" "RBS"] [Mode: No] => No
        DivisionOwnerCode in ["75"] [Mode: No] => No
        DivisionOwnerCode in ["9R"] [Mode: Yes]

```
                                        Weekend = Weekend [ Mode: No ]
                                                RepairSupplier = Yes [ Mode: No ] => No
                                                RepairSupplier = No [ Mode: Yes ]
                                                        Vitality in [ 1 ] [ Mode: Yes ] => Yes
                                                        Vitality in [ 2 3 4 5 ] [ Mode: No ] => No
                                        Weekend = Week [ Mode: Yes ] => Yes
AnalyzerCode in [ "024" ] [ Mode: No ]
        InventoryPosition <= 2,500 [ Mode: Yes ]
                #RR in [ 1 ] [ Mode: Yes ] => Yes
                #RR in [ 2 3 4 ] [ Mode: Yes ]
                        EOSAge <= 2,500 [ Mode: Yes ] => Yes
                        EOSAge > 2,500 [ Mode: No ] => No
        InventoryPosition > 2,500 [ Mode: No ]
                #RR in [ 1 2 ] [ Mode: No ] => No
                #RR in [ 3 4 ] [ Mode: Yes ]
                        BirthYearAge <= 4,500 [ Mode: No ] => No
                        BirthYearAge > 4,500 [ Mode: Yes ] => Yes
AnalyzerCode in [ "03V" ] [ Mode: No ]
        FractionRecentDemand <= 0,375 [ Mode: No ] => No
        FractionRecentDemand > 0,375 [ Mode: Yes ] => Yes
AnalyzerCode in [ "04A" ] [ Mode: Yes ]
        InventoryPosition <= 7,500 [ Mode: Yes ] => Yes
        InventoryPosition > 7,500 [ Mode: No ] => No
AnalyzerCode in [ "07B" ] [ Mode: Yes ]
        CriticalQuantity <= 1 [ Mode: Yes ] => Yes
        CriticalQuantity > 1 [ Mode: No ] => No
AnalyzerCode in [ "08L" ] [ Mode: Yes ]
        DivisionOwnerCode in [ "13" "26" "2D" "44" "48" "4S" "71" "75" "7G" "7H" "9R" "APL" "ASC" "BOI" "DUN" "G9" "LP" "LQ" "LU"
"LX" "MB" "MC" "MP" "MR" "MS" "MV" "OBB" "RBS" ] [ Mode: Yes ] => Yes
        DivisionOwnerCode in [ "MN" ] [ Mode: Yes ] => Yes
        DivisionOwnerCode in [ "MT" ] [ Mode: No ] => No
AnalyzerCode in [ "0C9" ] [ Mode: No ]
        ForecastedQuantity <= 15,750 [ Mode: No ] => No
        ForecastedQuantity > 15,750 [ Mode: Yes ] => Yes
AnalyzerCode in [ "0G3" ] [ Mode: No ]
        HistoricalActivity1Month <= 1,250 [ Mode: Yes ] => Yes
        HistoricalActivity1Month > 1,250 [ Mode: No ] => No
AnalyzerCode in [ "0G7" ] [ Mode: No ]
        #RR in [ 1 ] [ Mode: No ] => No
        #RR in [ 2 3 4 ] [ Mode: Yes ] => Yes
AnalyzerCode in [ "0W7" ] [ Mode: Yes ]
        HistoricalActivity2Years <= 29 [ Mode: Yes ] => Yes
        HistoricalActivity2Years > 29 [ Mode: No ] => No

Fold 2:
Weekend = Weekend [ Mode: Yes ]
        AnalyzerCode in [ "001" "004" "008" "013" "01X" "026" "04B" "05C" "09J" "09M" "0C6" "0G7" "0I3" "0V9" ] [ Mode: No ] => No
        AnalyzerCode in [ "005" ] [ Mode: No ]
                #RR in [ 1 ] [ Mode: No ] => No
                #RR in [ 2 3 4 ] [ Mode: Yes ]
                        Vitality in [ 1 ] [ Mode: Yes ] => Yes
                        Vitality in [ 2 3 4 5 ] [ Mode: No ] => No
        AnalyzerCode in [ "012" "018" "01A" "01B" "021" "023" "03D" "03V" "04A" "04J" "08F" "08Z" "09R" "09X" "0A7" "0M7" "0V7"
"0W7" "0W9" ] [ Mode: Yes ] => Yes
        AnalyzerCode in [ "014" ] [ Mode: No ]
                OpenOrderQuantity <= 3,500 [ Mode: Yes ] => Yes
                OpenOrderQuantity > 3,500 [ Mode: No ] => No
        AnalyzerCode in [ "01Z" "027" "03E" "03N" "04F" "085" "08C" "099" "09E" "09Y" "0C9" ] [ Mode: Yes ] => Yes
        AnalyzerCode in [ "024" ] [ Mode: No ]
                DivisionOwnerCode in [ "13" "26" "44" "48" "71" "75" "7G" "7H" "9R" "APL" "ASC" "BOI" "DUN" "LP" "LQ" "LU" "LX"
"MB" "MC" "MN" "MP" "MR" "MS" "MT" "MV" "OBB" "RBS" ] [ Mode: No ] => No
                DivisionOwnerCode in [ "2D" ] [ Mode: No ]
                        ForecastedQuantity <= 21,350 [ Mode: Yes ]
                                StockLevelRepair <= 1,500 [ Mode: Yes ]
                                        Vitality in [ 1 ] [ Mode: Yes ] => Yes
                                        Vitality in [ 2 3 4 5 ] [ Mode: Yes ]
                                                InventoryPosition <= 6,500 [ Mode: No ]
                                                        InventoryPosition <= 3,500 [ Mode: Yes ]
                                                                LifeCyclePosition <= 0,597 [ Mode: No ] => No
                                                                LifeCyclePosition > 0,597 [ Mode: Yes ] => Yes
                                                        InventoryPosition > 3,500 [ Mode: No ] => No
```

InventoryPosition > 6,500 [Mode: Yes] => Yes
StockLevelRepair > 1,500 [Mode: No] => No
ForecastedQuantity > 21,350 [Mode: No] => No
DivisionOwnerCode in ["4S"] [Mode: Yes] => Yes
DivisionOwnerCode in ["G9"] [Mode: No] => No
AnalyzerCode in ["030"] [Mode: No]
BirthYearAge <= 0,500 [Mode: Yes] => Yes
BirthYearAge > 0,500 [Mode: No] => No
AnalyzerCode in ["07B"] [Mode: Yes]
CriticalQuantity <= 1 [Mode: Yes] => Yes
CriticalQuantity > 1 [Mode: No] => No
AnalyzerCode in ["08L"] [Mode: Yes]
FirstStockAge in [0 1 2 3 4 5] [Mode: No] => No
FirstStockAge in [6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 22 23 53] [Mode: Yes] => Yes
AnalyzerCode in ["0G3"] [Mode: Yes]
Vitality in [1] [Mode: Yes] => Yes
Vitality in [2 3 4 5] [Mode: No] => No
AnalyzerCode in ["0U7"] [Mode: No]
ReturnRate <= 0,037 [Mode: No] => No
ReturnRate > 0,037 [Mode: Yes] => Yes
Weekend = Week [Mode: No]
AnalyzerCode in ["001"] [Mode: No]
RepairSupplier = Yes [Mode: Yes] => Yes
RepairSupplier = No [Mode: No] => No
AnalyzerCode in ["004" "013" "01X" "03D" "03E" "04J" "08F" "09R" "09X" "0I3" "0M7" "0V9"] [Mode: No] => No
AnalyzerCode in ["005"] [Mode: No]
OpenOrderQuantity <= 55 [Mode: No] => No
OpenOrderQuantity > 55 [Mode: Yes] => Yes
AnalyzerCode in ["008" "01A" "01B" "024" "026" "027" "03N" "03V" "04B" "04F" "05C" "07B" "08L" "08Z" "099" "09J" "09M" "0A7" "0C6" "0G3" "0G7" "0U7" "0W7" "0W9"] [Mode: No] => No
AnalyzerCode in ["012" "018" "01Z" "023" "085" "08C" "09E" "09Y" "0C9" "0V7"] [Mode: Yes] => Yes
AnalyzerCode in ["014"] [Mode: No]
InventoryPosition <= 31 [Mode: Yes]
InventoryPosition <= 6,500 [Mode: No] => No
InventoryPosition > 6,500 [Mode: Yes] => Yes
InventoryPosition > 31 [Mode: No] => No
AnalyzerCode in ["021"] [Mode: No]
DivisionOwnerCode in ["13" "26" "2D" "44" "48" "4S" "71" "7G" "7H" "APL" "ASC" "BOI" "DUN" "G9" "LP" "LQ" "LU" "LX" "MB" "MC" "MN" "MP" "MR" "MS" "MT" "MV" "OBB" "RBS"] [Mode: No] => No
DivisionOwnerCode in ["75"] [Mode: No] => No
DivisionOwnerCode in ["9R"] [Mode: Yes] => Yes
AnalyzerCode in ["030"] [Mode: No]
ForecastedQuantity <= 59,800 [Mode: No] => No
ForecastedQuantity > 59,800 [Mode: Yes] => Yes
AnalyzerCode in ["04A"] [Mode: No]
#RR in [1] [Mode: No] => No
#RR in [2 3 4] [Mode: Yes] => Yes

Fold 3:
AnalyzerCode in ["001"] [Mode: No]
HistoricalActivity2Years <= 2,500 [Mode: No] => No
HistoricalActivity2Years > 2,500 [Mode: No]
StockLevelRepair <= 24,500 [Mode: No]
HistoricalActivity1Month <= 0,875 [Mode: Yes]
Vitality in [1] [Mode: No]
InventoryPosition <= 6 [Mode: No] => No
InventoryPosition > 6 [Mode: Yes] => Yes
Vitality in [2 3 4 5] [Mode: Yes] => Yes
HistoricalActivity1Month > 0,875 [Mode: No] => No
StockLevelRepair > 24,500 [Mode: Yes] => Yes
AnalyzerCode in ["004" "008" "013" "01A" "01B" "01X" "01Z" "026" "027" "03N" "04B" "04F" "05C" "08Z" "09J" "09M" "0I3" "0U7" "0V9" "0W9"] [Mode: No] => No
AnalyzerCode in ["005"] [Mode: No]
FractionInventoryAvailable <= 0,183 [Mode: Yes] => Yes
FractionInventoryAvailable > 0,183 [Mode: No] => No
AnalyzerCode in ["012" "018" "023" "03D" "03E" "04J" "085" "08C" "08F" "09R" "09X" "09Y" "0A7" "0M7" "0V7" "0W7"] [Mode: Yes] => Yes
AnalyzerCode in ["014"] [Mode: Yes]
PSS <= 27,450 [Mode: Yes]
CriticalQuantity <= 0,500 [Mode: Yes]
FirstStockAge in [0 1 2 3 4] [Mode: Yes] => Yes

```
                                        FirstStockAge in [ 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 22 23 53 ] [ Mode: No ]
                                                BirthYearAge <= 7,500 [ Mode: No ] => No
                                                BirthYearAge > 7,500 [ Mode: Yes ] => Yes
                                CriticalQuantity > 0,500 [ Mode: Yes ] => Yes
                        PSS > 27,450 [ Mode: No ] => No
AnalyzerCode in [ "021" ] [ Mode: No ]
                        ShelflifeIndicator = Yes [ Mode: Yes ] => Yes
                        ShelflifeIndicator = No [ Mode: No ]
                                DivisionOwnerCode in [ "13" "26" "2D" "44" "48" "4S" "71" "7G" "7H" "APL" "ASC" "BOI" "DUN" "G9" "LP" "LQ" "LU"
"LX" "MB" "MC" "MN" "MP" "MR" "MS" "MT" "MV" "OBB" "RBS" ] [ Mode: No ] => No
                                DivisionOwnerCode in [ "75" ] [ Mode: No ]
                                        Weekend = Weekend [ Mode: No ]
                                                StockLevel <= 11,500 [ Mode: Yes ] => Yes
                                                StockLevel > 11,500 [ Mode: No ] => No
                                        Weekend = Week [ Mode: No ] => No
                                DivisionOwnerCode in [ "9R" ] [ Mode: Yes ]
                                        EOSAge <= 11,500 [ Mode: Yes ]
                                                RepairSupplier = Yes [ Mode: No ] => No
                                                RepairSupplier = No [ Mode: Yes ] => Yes
                                        EOSAge > 11,500 [ Mode: No ] => No
AnalyzerCode in [ "024" ] [ Mode: No ]
                        InventoryPosition <= 2,500 [ Mode: Yes ]
                                #RR in [ 1 ] [ Mode: Yes ] => Yes
                                #RR in [ 2 3 4 ] [ Mode: No ] => No
                        InventoryPosition > 2,500 [ Mode: No ]
                                DivisionOwnerCode in [ "13" "26" "44" "48" "71" "75" "7G" "9R" "APL" "ASC" "BOI" "DUN" "LP" "LQ" "LU" "LX" "MB"
"MC" "MN" "MP" "MR" "MS" "MT" "MV" "OBB" "RBS" ] [ Mode: No ] => No
                                DivisionOwnerCode in [ "2D" ] [ Mode: No ]
                                        Weekend = Weekend [ Mode: No ]
                                                PSS <= 82,350 [ Mode: No ]
                                                        OpenOrderIndicator = Yes [ Mode: No ]
                                                                CriticalQuantity <= 2,188 [ Mode: No ]
                                                                        #RR in [ 1 ] [ Mode: No ]
                                                                                CriticalQuantity <= 0,500 [ Mode: No ]
                                                                                        ReturnRate <= 0,281 [ Mode: No ]
                                                                                                RepairSupplier = Yes [ Mode:
No ] => No
                                                                                                RepairSupplier = No [ Mode:
No ]

                        ForecastedQuantity <= 126,100 [ Mode: Yes ] => Yes

                        ForecastedQuantity > 126,100 [ Mode: No ] => No
                                                                                        ReturnRate > 0,281 [ Mode: Yes ] => Yes
                                                                                CriticalQuantity > 0,500 [ Mode: No ] => No
                                                                        #RR in [ 2 3 4 ] [ Mode: No ] => No
                                                                CriticalQuantity > 2,188 [ Mode: Yes ] => Yes
                                                        OpenOrderIndicator = No [ Mode: No ] => No
                                                PSS > 82,350 [ Mode: Yes ] => Yes
                                        Weekend = Week [ Mode: No ] => No
                                DivisionOwnerCode in [ "4S" ] [ Mode: Yes ] => Yes
                                DivisionOwnerCode in [ "7H" ] [ Mode: No ] => No
                                DivisionOwnerCode in [ "G9" ] [ Mode: No ]
                                        ReturnRate <= 0,035 [ Mode: Yes ] => Yes
                                        ReturnRate > 0,035 [ Mode: No ] => No
AnalyzerCode in [ "030" ] [ Mode: No ]
                        BirthYearAge <= 0,500 [ Mode: Yes ] => Yes
                        BirthYearAge > 0,500 [ Mode: No ] => No
AnalyzerCode in [ "03V" ] [ Mode: No ]
                        FractionRecentDemand <= 0,375 [ Mode: No ] => No
                        FractionRecentDemand > 0,375 [ Mode: Yes ] => Yes
AnalyzerCode in [ "04A" ] [ Mode: Yes ]
                        OpenOrderIndicator = Yes [ Mode: No ] => No
                        OpenOrderIndicator = No [ Mode: Yes ]
                                HistoricalActivity2Years <= 12,500 [ Mode: Yes ] => Yes
                                HistoricalActivity2Years > 12,500 [ Mode: No ] => No
AnalyzerCode in [ "07B" ] [ Mode: Yes ]
                        CriticalQuantity <= 1 [ Mode: Yes ] => Yes
                        CriticalQuantity > 1 [ Mode: No ] => No
AnalyzerCode in [ "08L" ] [ Mode: No ]
```

DivisionOwnerCode in ["13" "26" "2D" "44" "48" "4S" "71" "75" "7G" "7H" "9R" "APL" "ASC" "BOI" "DUN" "G9" "LP" "LQ" "LU" "LX" "MB" "MC" "MP" "MR" "MS" "MV" "OBB" "RBS"] [Mode: Yes] => Yes
DivisionOwnerCode in ["MN"] [Mode: Yes] => Yes
DivisionOwnerCode in ["MT"] [Mode: No] => No
AnalyzerCode in ["099" "09E"] [Mode: No] => No
AnalyzerCode in ["0C6"] [Mode: No]
#RR in [1] [Mode: No] => No
#RR in [2 3 4] [Mode: Yes] => Yes
AnalyzerCode in ["0C9"] [Mode: No]
ForecastedQuantity <= 15,750 [Mode: No] => No
ForecastedQuantity > 15,750 [Mode: Yes] => Yes
AnalyzerCode in ["0G3"] [Mode: No]
Vitality in [1] [Mode: Yes]
CriticalQuantity <= 1,500 [Mode: Yes] => Yes
CriticalQuantity > 1,500 [Mode: No] => No
Vitality in [2 3 4 5] [Mode: No] => No
AnalyzerCode in ["0G7"] [Mode: No]
#RR in [1] [Mode: No] => No
#RR in [2 3 4] [Mode: Yes] => Yes

Fold 4:
AnalyzerCode in ["001" "004" "008" "013" "01A" "01B" "01Z" "026" "030" "03V" "04B" "08L" "099" "09J" "09M" "0G3" "0I3" "0U7" "0V9" "0W9"] [Mode: No] => No
AnalyzerCode in ["005"] [Mode: No]
HistoricalActivity1Month <= 19,500 [Mode: No] => No
HistoricalActivity1Month > 19,500 [Mode: Yes] => Yes
AnalyzerCode in ["012" "018" "023" "03D" "03E" "04A" "04J" "07B" "08C" "08F" "08Z" "09E" "09R" "09X" "0A7" "0M7"] [Mode: Yes] => Yes
AnalyzerCode in ["014"] [Mode: No]
InventoryPosition <= 44,500 [Mode: Yes]
DivisionOwnerCode in ["13" "26" "44" "48" "4S" "71" "7G" "7H" "APL" "ASC" "BOI" "DUN" "G9" "LP" "LQ" "LU" "LX" "MB" "MC" "MN" "MP" "MR" "MS" "MT" "MV" "OBB" "RBS"] [Mode: Yes] => Yes
DivisionOwnerCode in ["2D" "75"] [Mode: Yes] => Yes
DivisionOwnerCode in ["9R"] [Mode: Yes]
FractionInventoryAvailable <= 0,935 [Mode: No] => No
FractionInventoryAvailable > 0,935 [Mode: Yes]
FirstStockAge in [0 1 2 3 4] [Mode: Yes] => Yes
FirstStockAge in [5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 22 23 53] [Mode: No] => No
InventoryPosition > 44,500 [Mode: No] => No
AnalyzerCode in ["01X" "027" "03N" "04F" "085" "09Y"] [Mode: No] => No
AnalyzerCode in ["021"] [Mode: No]
ShelflifeIndicator = Yes [Mode: Yes] => Yes
ShelflifeIndicator = No [Mode: No] => No
AnalyzerCode in ["024"] [Mode: No]
Weekend = Weekend [Mode: No]
FractionRecentDemand <= 0,450 [Mode: No] => No
FractionRecentDemand > 0,450 [Mode: Yes] => Yes
Weekend = Week [Mode: No] => No
AnalyzerCode in ["05C"] [Mode: No]
ReturnRate <= 0,539 [Mode: No] => No
ReturnRate > 0,539 [Mode: Yes] => Yes
AnalyzerCode in ["0C6"] [Mode: No]
FirstStockAge in [0 1 2 3 4] [Mode: No] => No
FirstStockAge in [5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 22 23 53] [Mode: Yes] => Yes
AnalyzerCode in ["0C9"] [Mode: No]
ForecastedQuantity <= 15,750 [Mode: No] => No
ForecastedQuantity > 15,750 [Mode: Yes] => Yes
AnalyzerCode in ["0G7"] [Mode: No]
#RR in [1] [Mode: No] => No
#RR in [2 3 4] [Mode: Yes] => Yes
AnalyzerCode in ["0V7"] [Mode: Yes]
EOSAge <= 2,500 [Mode: No] => No
EOSAge > 2,500 [Mode: Yes] => Yes
AnalyzerCode in ["0W7"] [Mode: Yes]
PSS <= 5,075 [Mode: Yes] => Yes
PSS > 5,075 [Mode: No] => No

Fold 5:
AnalyzerCode in ["001" "004" "008" "013" "01A" "01B" "01X" "01Z" "021" "026" "027" "030" "03N" "04B" "04F" "08Z" "099" "09J" "09M" "0C9" "0G7" "0I3" "0U7" "0V9" "0W9"] [Mode: No] => No
AnalyzerCode in ["005"] [Mode: No]

```
FractionInventoryAvailable <= 0,183 [ Mode: Yes ] => Yes
FractionInventoryAvailable > 0,183 [ Mode: No ]
Weekend = Weekend [ Mode: No ]
#RR in [ 1 ] [ Mode: No ] => No
#RR in [ 2 3 4 ] [ Mode: Yes ]
ForecastedQuantity <= 6,500 [ Mode: No ] => No
ForecastedQuantity > 6,500 [ Mode: Yes ] => Yes
Weekend = Week [ Mode: No ] => No
AnalyzerCode in [ "012" "018" "023" "03E" "04J" "085" "08F" "09E" "09R" "09X" "09Y" "0A7" "0M7" "0V7" ] [ Mode: Yes ] => Yes
AnalyzerCode in [ "014" ] [ Mode: No ]
PSS <= 27,450 [ Mode: Yes ] => Yes
PSS > 27,450 [ Mode: No ] => No
AnalyzerCode in [ "024" ] [ Mode: No ]
EOSAge <= 0,500 [ Mode: Yes ]
PSS <= 14,150 [ Mode: Yes ] => Yes
PSS > 14,150 [ Mode: No ] => No
EOSAge > 0,500 [ Mode: No ]
DivisionOwnerCode in [ "13" "26" "44" "48" "71" "75" "7G" "7H" "9R" "APL" "ASC" "BOI" "DUN" "LP" "LQ" "LU" "LX"
"MB" "MC" "MN" "MP" "MR" "MS" "MT" "MV" "OBB" "RBS" ] [ Mode: No ] => No
DivisionOwnerCode in [ "2D" ] [ Mode: No ]
#RR in [ 1 2 ] [ Mode: No ]
#RR in [ 1 ] [ Mode: No ]
Vitality in [ 1 ] [ Mode: No ] => No
Vitality in [ 2 3 4 5 ] [ Mode: No ]
Weekend = Weekend [ Mode: No ]
CriticalQuantity <= 2,200 [ Mode: No ] => No
CriticalQuantity > 2,200 [ Mode: Yes ] => Yes
Weekend = Week [ Mode: No ]
HistoricalActivity1Month <= 7,750 [ Mode: No ] => No
HistoricalActivity1Month > 7,750 [ Mode: No ]
ForecastedQuantity <= 984,100 [ Mode: Yes ] => Yes
ForecastedQuantity > 984,100 [ Mode: No ] => No
#RR in [ 2 ] [ Mode: No ]
PSS <= 0,850 [ Mode: Yes ] => Yes
PSS > 0,850 [ Mode: No ] => No
#RR in [ 3 4 ] [ Mode: Yes ] => Yes
DivisionOwnerCode in [ "4S" ] [ Mode: Yes ] => Yes
DivisionOwnerCode in [ "G9" ] [ Mode: No ] => No
AnalyzerCode in [ "03D" "08C" ] [ Mode: No ] => No
AnalyzerCode in [ "03V" ] [ Mode: No ]
FractionRecentDemand <= 0,375 [ Mode: No ] => No
FractionRecentDemand > 0,375 [ Mode: Yes ] => Yes
AnalyzerCode in [ "04A" ] [ Mode: Yes ]
HistoricalActivity2Years <= 12,500 [ Mode: Yes ] => Yes
HistoricalActivity2Years > 12,500 [ Mode: No ] => No
AnalyzerCode in [ "05C" ] [ Mode: No ]
ReturnRate <= 0,362 [ Mode: No ] => No
ReturnRate > 0,362 [ Mode: Yes ] => Yes
AnalyzerCode in [ "07B" ] [ Mode: Yes ]
CriticalQuantity <= 1 [ Mode: Yes ] => Yes
CriticalQuantity > 1 [ Mode: No ] => No
AnalyzerCode in [ "08L" ] [ Mode: No ]
DivisionOwnerCode in [ "13" "26" "2D" "44" "48" "4S" "71" "75" "7G" "7H" "9R" "APL" "ASC" "BOI" "DUN" "G9" "LP" "LQ" "LU"
"LX" "MB" "MC" "MP" "MR" "MS" "MV" "OBB" "RBS" ] [ Mode: Yes ] => Yes
DivisionOwnerCode in [ "MN" ] [ Mode: Yes ] => Yes
DivisionOwnerCode in [ "MT" ] [ Mode: No ] => No
AnalyzerCode in [ "0C6" ] [ Mode: No ]
FractionRecentDemand <= 0,163 [ Mode: No ] => No
FractionRecentDemand > 0,163 [ Mode: Yes ] => Yes
AnalyzerCode in [ "0G3" ] [ Mode: No ]
Vitality in [ 1 ] [ Mode: Yes ]
CriticalQuantity <= 1,500 [ Mode: Yes ] => Yes
CriticalQuantity > 1,500 [ Mode: No ] => No
Vitality in [ 2 3 4 5 ] [ Mode: No ]
MinimumLeadTime <= 15,900 [ Mode: No ] => No
MinimumLeadTime > 15,900 [ Mode: Yes ] => Yes
AnalyzerCode in [ "0W7" ] [ Mode: Yes ]
InventoryPosition <= 57 [ Mode: Yes ] => Yes
InventoryPosition > 57 [ Mode: No ] => No
```