Improving the Supply Chain using Artificial Intelligence

Sofia Danielsson, Emma Ekström
Improving the Supply Chain using Artificial Intelligence

Sofia Danielsson
inel3sda@student.lu.se

Emma Ekström
inel3eek@student.lu.se

January 12, 2018

Master’s thesis work carried out at
the Department of Computer Science, Lund University.

Supervisor: Elin Anna Topp, elin_anna.topp@cs.lth.se
Examiner: Jacek Malec, jacek.malec@cs.lth.se
Abstract

Though artificial intelligence, AI, has been around for the last half-century and has greatly improved performances in many industries, the technology has not been fully exploited in supply chain management, SCM. The flows in today’s supply chains are usually fast-moving and dynamic, but are often handled manually. This can inhibit the organizations’ capability to adapt to the quick changes in demand. To be able to quickly respond to these changes, cooperation between the purchasing company and the supplier is needed. Despite this, many companies are afraid to share information with each other, obstructing cooperation. The use of AI could provide a dynamic solution to these problems.

The overall objective of this project was to examine how AI could be applied to SCM and what benefits this could enclose. By interviewing people working with SCM, problems within the area and desired solutions could be mapped. As there existed a desire for a system that predicts whether an order will arrive on time, a prototype for such a system was implemented. By analyzing data through experimental testing, using the data mining tool Weka, a suitable AI model could be obtained.

It has been concluded that the performance of the model depends on some key conditions. The most crucial factor is the presence of patterns in the supplier’s behavior. If these conditions are taken into consideration, AI can be used within SCM to provide valuable information. The positive effects of displaying this information can benefit both the purchasing company and the supplier.

Keywords: Artificial intelligence, Probabilistic reasoning, Bayesian networks, Supply chain management, Order handling
Acknowledgments

This master thesis was initiated by our collaborative partner, PipeChain Group AB. It would not have been possible to carry out the analyses made during this project without their contributions. We would like to thank our supervisors at PipeChain, Oskar Präntare and Klas Turesson, for their support and guidance.

The information gained from interviews, conducted with some of PipeChain’s customers and consultants, has given a great contribution to the conclusions made. Therefore we want to thank the companies and employees that participated in these.

We would also like to thank our supervisor at LTH, Elin Anna Topp, who has given us great input, which has improved the quality of the project considerably.
# Contents

1 Introduction 7  
1.1 Background 7  
1.2 Vocabulary 8  
1.2.1 Supply chain related words 8  
1.2.2 Computer scientific words 9  
1.2.3 Abbreviations 9  
1.3 Delimitation 9  
1.4 Objective and problem formulation 10  
1.5 Collaboration with PipeChain Group AB 10  
1.6 Contribution statement 11  
1.6.1 Responsibility distribution 11  
1.6.2 Contributions 11  
1.7 Method 11  
1.7.1 Interview phase 12  
1.7.2 Test phase 13  
1.7.3 Development phase 14  
1.7.4 Evaluation phase 14  
1.8 Outline of the report 14  

2 Problem analysis 15  
2.1 Known problems and wanted features 15  
2.2 Initial prototype description 17  
2.2.1 Classification 17
3 Frame of reference

3.1 Supply Chain Management ........................................ 19
   3.1.1 The concept of SCM ........................................ 19
   3.1.2 The order process .......................................... 20

3.2 Artificial Intelligence ........................................... 21
   3.2.1 Absolute independence vs. conditional independence .... 22
   3.2.2 Bayesian networks .......................................... 22
   3.2.3 Learning algorithms that manipulate probabilities ...... 23
   3.2.4 Decision trees ............................................. 24
   3.2.5 Validation .................................................. 25

3.3 Related work ..................................................... 26

4 Applying AI to order placement ..................................... 27

4.1 Weka ............................................................. 27
   4.1.1 Algorithms .................................................. 27

4.2 Data cleansing ..................................................... 31

4.3 Attribute selection ............................................... 32
   4.3.1 Base attributes ............................................. 32
   4.3.2 Experimental testing ....................................... 33

4.4 Choice of algorithm ............................................. 40

4.5 Refining the chosen algorithm ................................... 41
   4.5.1 Transforming multiple classes to binary .................. 45
   4.5.2 Choosing search algorithm ................................ 45
   4.5.3 Discretization of continuous variables .................... 45
   4.5.4 Handle class-unbalanced data ............................. 46
   4.5.5 Additional attributes ...................................... 47
   4.5.6 Grouping suppliers ........................................ 51
   4.5.7 Additional features ........................................ 52
5 Final prototype and results
  5.1 Specification ..................................................... 55
    5.1.1 User interface ............................................ 56
    5.1.2 Updating the model ...................................... 58
  5.2 Evaluation ...................................................... 59
    5.2.1 Self-awareness of the algorithm ......................... 62
    5.2.2 The robustness of the prototype ......................... 65
    5.2.3 Overfitting ............................................... 66

6 Discussion ......................................................... 67
  6.1 Difficulties ..................................................... 67
  6.2 Seasonal variations ........................................... 68
  6.3 Limitations of the prototype ................................ 68
  6.4 The benefits of the prototype ............................... 69
  6.5 Suitable conditions of use ................................... 70
  6.6 Risks ............................................................ 71

7 Conclusions ........................................................ 72

8 References .......................................................... 75
  Articles ............................................................. 75
  Books ................................................................. 76
  Websites ............................................................ 76
1 Introduction

This chapter serves as an introduction to the report. A short background on the problem at hand is presented and thereafter the objective, problem formulation and delimitation of the project are stated.

1.1 Background

Artificial intelligence, AI, has been around for the last half-century and has greatly improved performances in many industries, for example by providing decision support. Despite its great potential and proven record, the technology has been used to a limited extent in supply chain management, SCM (Min 2008).

The notion supply chain encapsulates the collection of organizations that are united through flows of services, products, information or finances from the source to the ultimate consumer. The flows in today’s supply chains are often fast-moving and dynamic. Despite this, the flows are in many cases handled in old systems, where a lot of job is still done manually. This can inhibit the organizations’ capability to adapt to the quick changes in demand. Many companies are also afraid to share company-specific information with other parts of the supply chain, since the information could be used in ill-meaning ways (M. Cooper, Lambert, and Pagh 1997). This obstructs the cooperation between organizations.

There are advanced technologies that could be applied to SCM, for example technologies within AI. Especially learning algorithms, which can find patterns and adapt to different situations could be suitable for the changing flows of today’s supply chains. They could see patterns that humans have a hard time finding and thereby help humans understand why a certain outcome took place. Many companies have access to a lot of data that could potentially be used to solve problems they are facing. But it can be difficult to process all data and it can take a lot of time to analyze, indicating that the solution to a problem will be quite static. If using this valuable data to build a learning scheme, some of the problems currently occurring in SCM could potentially be reduced, and the solution would be more dynamic.
1.2 Vocabulary

In this section words and abbreviations that are used throughout the report are defined. These are shown in italic the first time they are mentioned in the report.

1.2.1 Supply chain related words

<table>
<thead>
<tr>
<th>Word</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supplier</td>
<td>A company selling items to another company.</td>
</tr>
<tr>
<td>Purchasing company</td>
<td>A company buying items from a supplier.</td>
</tr>
<tr>
<td>Supply chain</td>
<td>A group of organizations connected through the flows of goods, services, information or finances. One organization is considered to be part of the supply chain if it in some way contributes to the process of producing a complete product. A part of the supply chain can be represented by for example a supplier of raw materials or components, a manufacturing company or a transport company.</td>
</tr>
<tr>
<td>Forecast accuracy</td>
<td>The share of orders that agree with the forecasted demand.</td>
</tr>
<tr>
<td>Delivery precision</td>
<td>The supplier’s capability to deliver the right amount of items at the right time.</td>
</tr>
<tr>
<td>Traditional order placement or purchasing</td>
<td>Defined in this report as when a customer places an order for materials at a supplier. This is further explained in section 3.1.2.</td>
</tr>
<tr>
<td>Vendor-managed inventory</td>
<td>When a supplier commits to keeping the customer inventory level between a predefined upper and lower bound and automatically refills the inventory when needed. This means the customer is not placing any orders, in contrast to traditional order placement.</td>
</tr>
<tr>
<td>Key performance indicator</td>
<td>A set of measurable values that an organization uses to survey their performance over time.</td>
</tr>
<tr>
<td>Lead time</td>
<td>The time between the placement of the order and its delivery.</td>
</tr>
</tbody>
</table>
1.2.2 Computer scientific words

<table>
<thead>
<tr>
<th>Word</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time complexity</td>
<td>The quantification of the time consumption of an algorithm given its input values.</td>
</tr>
<tr>
<td>Instance</td>
<td>One data sample from a dataset.</td>
</tr>
<tr>
<td>Attribute</td>
<td>A value that an instance can take. For example &quot;name&quot; could be an attribute and the name of one instance could be &quot;Smith&quot;.</td>
</tr>
<tr>
<td>Greedy search</td>
<td>An algorithm that searches greedily for a solution. This means it takes small steps in the space of possible solutions, where each step optimizes an underlying criterion. These algorithms are called greedy since they choose the best local alternative in each step, not necessarily yielding an optimal solution.</td>
</tr>
</tbody>
</table>

1.2.3 Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI</td>
<td>Artificial Intelligence</td>
</tr>
<tr>
<td>SCM</td>
<td>Supply Chain Management</td>
</tr>
<tr>
<td>VMI</td>
<td>Vendor Managed Inventory</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>KPI</td>
<td>Key Performance Indicator</td>
</tr>
</tbody>
</table>

1.3 Delimitation

Since the SCM field is very wide and the available time for this project was limited, a delimitation was necessary. A big part of the contact between different organizations in the supply chain consists of purchasing, which made this an intriguing starting point. Purchasing can be made on three different levels; strategic, tactical and operational. These are described in 3.1.1. In this project, operational purchasing was the main focus, which is centralized around order handling. Henceforth when purchasing is mentioned, it is operational purchasing that is discussed if nothing else is stated. Many companies have problems regarding how to place their orders to receive the requested item in the right amount and at the right time. Therefore this report focused on how AI can be used to minimize problems during order placement.

The main focus of the report was probabilistic reasoning, since it was believed to be most suitable for the data at hand. Neural networks are very popular, but are
not discussed in this report. The system in mind should give purchasers valuable information to base their decisions upon. A neural network works like a black box, not showing at all what it has based its classifications upon, making it ill-fitted for such a system. Meanwhile, probabilistic methods can show what an outcome is based upon and give a probability for it being true.

The report only looks upon available internal data, and does not consider any external factors that might affect the result. When writing this report, it was assumed that the reader has basic knowledge within computer science.

1.4 Objective and problem formulation

The overall objective of this project is to examine how AI can be applied to SCM and if this can lead to advantages for both the supplier and the purchasing company. The implementation of a prototype, where predictions based on customer data are made, is also a part of the main objective. The following questions will be answered:

1. What problems are present in today’s order process and which of them are possible to solve with the aid of artificial intelligence?
2. What benefits can the supplier and the purchasing company obtain by using a prototype like the one produced during this project?
3. Which data and what amount of data is needed to make reliable predictions of the flow of materials in the supply chain?

1.5 Collaboration with PipeChain Group AB

This project was conducted in collaboration with PipeChain Group AB. PipeChain operates within SCM. They develop and sell a software product that their customers can use in order to control the supply chain. PipeChain is a cloud service, which enables an overall picture of the supply chain and an early detection of delivery issues. It also gives the customer and supplier access to the same *key performance indicators*, KPI, as to make the collaboration smoother (PipeChain 2016). In order handling, PipeChain offers services for both traditional purchasing and *vendor-managed inventory*, VMI. When it comes to VMI, PipeChain’s software calculates the maximum and minimum acceptable values for the inventory and suggests when the supplier should ship goods in order to remain in between these levels. When and if the supplier confirms these suggestions, an order is automatically placed. It is most common to *place traditional orders* in the system, rather than using VMI. The majority of PipeChain’s customers are producing companies and mostly do the final steps of the production. All are business-to-business companies, i.e., they only sell to companies and not to private individuals.
Since many companies perform SCM using old technologies, it was believed to be difficult to find companies that were willing to spend time on analyzing the potential benefits from using AI. By working together with PipeChain, easier access to people willing to discuss this matter was obtained. This collaboration also gave access to valuable data that could be analyzed in order to see how much and which data that was needed in order to apply AI to SCM.

1.6 Contribution statement

In this section it is described how the responsibilities of the work have been distributed and what contributions this project could provide for the business and research communities.

1.6.1 Responsibility distribution

As Emma specializes within software-intensive systems, she has had the main responsibility for programming tasks, including programming the application programming interface, API, representing the prototype.

Sofia specializes in supply chain management, and has therefore had the main responsibility for supply-chain-related tasks. She has also had the overall responsibility for the interviews.

Overall we wanted to do as much as possible together. Therefore all testing and analyzing of data have been conducted in collaboration, as well as the formulation of all discussion and conclusion material.

1.6.2 Contributions

If the prototype is regarded as successful, the idea can be further developed and integrated as a part of PipeChain’s current system. This means the product could possibly be used by PipeChain’s customers, which could alter the usage of AI within SCM. For example the usage of a system like our prototype could simplify the cooperation between organizations, without the need to share more information with each other.

This report suggests many interesting areas that could be further investigated. Section 7 describes some ideas on how this research could be developed further.

1.7 Method

Our method followed a four-phase approach: interview, test, development and evaluation, see figure 1.1.
During the interview phase, problems and potential improvement areas could be found, and the output from this stage was a prototype description. The test phase concerned finding the right algorithm and attributes for the prototype. The output of the development phase was the prototype itself and it was evaluated during the last phase. Each phase will be described in an individual section below.

### 1.7.1 Interview phase

Interviews with selected suppliers and purchasers and some of PipeChain’s own consultants were performed. These interviews could reveal relevant issues to look at and give an insight into the supply chain processes. The information that was gained during the interviews was later used to develop requirements for the prototype. These requirements are presented in chapter 2.

The number of informants was assumed to be limited. Interviews with a few suppliers and purchasers were therefore considered to be the best way of obtaining knowledge instead of conducting a more general research. Interviews give a deep knowledge and are usually conducted with few people. Interviews also give an ability to compare experiences from the interviewees with the data at hand (Denscombe 2000, p. 132-133), which could make it easier to find patterns.

#### Qualifying criteria and chosen companies

As mentioned in section 1.5, finding companies willing to spend time on being interviewed can be difficult. It is also important that the company’s order process is digitized and the information about the orders is saved for some time, since the customer then might have some understanding of the patterns in their order placement. The qualifying criteria for participation in the study are therefore:

1. Companies must be customers to PipeChain Group AB.
2. Companies must have enough collaboration with their suppliers to show sufficient data, both in time and amount.
1.7. METHOD

1.7.2 Test phase

After deciding on which features should be implemented in the prototype based on the conducted interviews, a test phase was initiated. PipeChain’s data was analyzed in order to investigate how it could be used in the implementation. In order to structure the test phase, a mind map of all the important decisions we had to make was drawn. The mind map can be seen in figure 1.2.

![Mind map](image)

Figure 1.2: The mind map used when looking for the right algorithm and settings.

A great part of the testing involved finding which attributes to use in the analysis. Three different AI algorithms were also tested against each other, to make an evaluation of different classification techniques. As the system in mind mainly focuses on providing information, rather than giving direction of which decision to make, it is important to be able to give a probability for the given prediction. This could add an additional layer of information to the purchasers, which could allow them to make well-based decisions. The hypothesis was that Bayesian networks would be the ultimate choice, since probabilities can easily be obtained when using this algorithm (Mitchell 1997, p. 155). Since Bayesian networks were believed to be the most suitable for the problem at hand, more information about them will be presented in the Frame of Reference chapter.

Decision trees could also be suitable for the system in mind. Therefore these were investigated further, in order to make a well-founded decision of which algorithm to use. We chose to investigate three algorithms whereof two use Bayesian networks and one uses decision trees. The three algorithms chosen were a traditional Bayes optimal classifier, a naïve Bayes classifier and a decision tree classifier called J48.
The result of this testing can be seen in 4.3.2. When one algorithm had been chosen, refinement of that algorithm was performed. There are many settings that can be made to the different algorithms, as can be seen in section 4.1.1. When altering a setting did not produce any difference in the accuracy, this was not recorded in order to reduce the length of the report. Finally, additional features were evaluated. The ultimate scenario would be to try all the settings on all available customers in PipeChain’s system. Since the time frame was limited, the focus has been on one customer and a couple of its suppliers. Random samples have been made with other customers and suppliers, to check the accuracy of the algorithm.

1.7.3 Development phase

Ultimately, the prototype was implemented during the development phase, showing how AI can be used within SCM and the order process. In this phase, the API of the data mining tool Weka was used in order to program the prototype. Weka is further described in section 4.1. The implementation consists of an API that converts PipeChain’s order line data to a structure that is recognized by Weka, builds models based on training data and classifies instances. The algorithm, attributes and settings chosen can be seen in section 5.1.

1.7.4 Evaluation phase

As to know how trustworthy the algorithm used by the prototype was, it was analyzed in the evaluation phase. The subjects evaluated were accuracy, self-awareness, robustness and overfitting. The results from this evaluation are presented in section 5.2.

1.8 Outline of the report

In the next chapter, issues in today’s purchasing are identified based on the conducted interviews, and the requirements of the prototype aiming at solving these problems are presented. This is followed by a frame of reference, where SCM and AI theory relevant to the suggestions made in chapter 2 are described. Other publications inspiring our work are also described here. In chapter 4 the experimental testing of different AI algorithms is presented. All efforts to obtain a better accuracy are recorded here. In chapter 5, the accuracy, error rate and other values describing the final system are shown. At the end of the report, a discussion and conclusion are presented, as well as suggestions to how others could keep investigating this area.
This chapter presents a summary of the information gained during the interviews and a prototype description describing the wanted features. A more detailed description of what the interviewed customers to PipeChain had to say is presented in Appendix C. A structured interview, where all questions and answering alternatives are given seemed too narrow for this purpose. Hence, semi-structured interviews were conducted. This is the most common form of qualitative studies (Nicholls 2017). Both field notes and sound recording were made in order to capture as much as possible during the interviews. The contents in Appendix C was later sent to the interviewees, in order to validate that the information was correctly interpreted (Denscombe 2000, p. 158). The interview guide with questions for customers can be found in Appendix A and for consultants in Appendix B.

2.1 Known problems and wanted features

We interviewed two customers representing purchasing companies and one customer representing a supplier. One of the purchasing companies only used traditional order placement and the other one used a mix of both traditional and VMI. The supplier company used VMI towards their customers. After having conducted the interviews, several conclusions could be drawn. All interviewed companies had some kind of problem with their suppliers or customers. Typical problems for purchasing companies in traditional order placement are suppliers not confirming the order, not delivering as they have confirmed or keeping information about problems hidden, as to not lose face. The interviewees believed predictions could help their companies’ order placement, by for example getting an early warning about a supplier who has a tendency to deliver late. A prediction showing how many orders that could be affected and how late they would be, would be even better. Also an action proposal showing whether the orders should be placed earlier, or in a different quantity would be beneficial. The biggest problem for the supplier using VMI was the low forecast accuracy of its customers, which made it hard for the supplier to plan its deliveries.

Four consultants were interviewed. They work as project managers as well as application consultants. They have all worked at PipeChain for quite a long time and have many years of experience within SCM. All consultants seemed positive to
a feature where the probability of an order arriving on time is shown when placing or changing an order. This could help the production planning for purchasing companies. If there is a high probability of the order arriving on time, they might be willing to plan their production according to this and otherwise they might want to reschedule. One of the consultants expressed a concern that the customers would only wait until the supplier confirms the order, and would therefore not take any actions when the probability is given. He therefore wanted to see another approach, where the probability of a supplier delivering on time would be shown a couple of weeks in advance, for example "Supplier x will only be able to deliver 70% of the ordered items on time in November." Customers might wait for a confirmation in cases when a supplier always confirms the orders in good time and delivers accordingly. One problem some purchasing companies seemed to have though was that some suppliers almost never made an order confirmation. In these cases, predictions on the outcome of an order could be of great value. This could help the customers to know when they should take action and maybe alter their production plan. As information from the purchasing companies showed, some also struggled with suppliers that did confirm the orders, but delivered on the wrong date anyway. Therefore it could also be interesting to give the purchasing companies a new prediction when the supplier has confirmed the order. This kind of prediction could give an indication on which confirmations to trust.

The system could be of importance for more than just typical production companies. Reliable deliveries are critical in the construction industry, since no inventory is held. During the interviews, the construction industry was pointed out as lagging behind when it comes to digitalization, though. There is a tradition within this industry to order via telephone calls, which makes the order hard to trace and follow up on. The automotive industry was also pointed out as a industry where these kind of predictions could be suitable, since its technical use is highly developed.

The consultants defined one of the major problems with today’s purchasing to be that many operative purchasers have limited understanding about the tactical and strategic approaches of the company. They often lack the knowledge about cause and effect, which could be due to not having a proper educational background. People working with cause and effect often have a position which places them far away from the daily activities, i.e., the operative buying. This means there is a big gap between the people needing the information and the ones looking at it. The operative purchasers also tend to either place the order too late since they do not know or understand the agreed upon lead time or too early, leading to the need to change the orders later, making the suppliers confused and ill-prepared. Many customers experienced some seasonal variations or campaigning affecting their business, which could lead to issues.

Another problem that was pointed out was how some people use the KPIs. Instead of trying to find the root cause when the KPI deteriorate, some purchasing companies tend to demand an answer to why the supplier has not performed their job. Some people have a hard time understanding that their actions affect the supplier’s delivery precision. For instance, an order placement forecast with very low accuracy could affect the supplier very much, as the orders the customer places badly matches the
2.2. Initial prototype description

As the interviews confirmed that order placement is an interesting area to look at with quite many problems, this strengthened our belief that the implemented prototype should aim at solving problems in this area. The prototype should make predictions for single order lines and not entire orders since the data can differ a lot between different order lines. The first version should be predicting the probability of different delivery time outcomes of an order line, since this seemed interesting to all interviewees. The predictions could be divided into different intervals, as to represent if the order will be early, on time or late. Hence, the problem was considered to be a classification problem. There was also an interest in getting action proposals. Such proposals would have a minor effect on the purchaser’s behavior if not knowing what probability there is for the wanted order to arrive on time in the first place. Therefore the first step will be to predict when an ordered item will arrive. The focus will be on traditional order placement, since more problems were found in this area during the interviews. The interviewees also had problems with VMI, but it seemed to be mostly bad forecast accuracies that caused the problems. An algorithm looking at these accuracies could probably be integrated with PipeChain’s current VMI product, but this was not investigated further during this project.

It could also be interesting to predict whether an order line item will arrive in the correct quantity, since some companies have greater issues regarding this than regarding delivery time. This could serve as a second version of the product.

As mentioned above it might also be helpful to offer the customers a new prediction when the supplier has confirmed the order. This could be done both with respect to delivery time and delivery quantity.

Several interviewees also requested some kind of early indicator that a supplier will probably begin to deliver badly. One way to do this could be by also including real-time data from the surrounding world. This was considered out of the scope of this project, though. Nevertheless, we also interpreted this response as an indicator that the interviewees would benefit from some kind of compilation of all the predictions that have been made during a certain time period for a certain supplier. Since there are often many different purchasers placing orders, they might have a hard time seeing overall patterns if they only get to see the predicted outcomes of individual orders. Therefore a later version of the prototype could include a function that compiles the predictions into a predicted delivery precision value.

2.2.1 Classification

Based on the requirements of the prototype it was concluded that a classification problem was at hand. When solving a classification problem, the algorithm tries to...
assign instances to one of the possible classes. As a starting point, there were four classes the algorithm could assign each instance to. All classes are time intervals, and these are based on the information given by the interviews. In this case, the classification is therefore a kind of prediction, telling when an item is likely to arrive. All interviews indicated that an ordered item is usually perceived as on time if it arrives on the demanded delivery date or a few days before, but not if it arrives later. Many companies also order the material with some days margin, as to not run out of material if the order would be delayed. That means that a delayed order will have the largest impact on the business when arriving later than the margin taken. As to represent this, we chose to have two intervals for late orders. The following intervals were chosen:

- More than 2 days early.
- On time, which is defined as either 1-2 days early or on the correct delivery day.
- 1-10 days late.
- More than 10 days late.

These classes could of course be adapted to each customer, as to reflect the intervals that they consider to be of most importance. Since the prototype was designed to make predictions for a single order line, an instance was represented by one order line.
3 | Frame of reference

In this chapter, theoretic material that represents a base for the thesis is presented. It contains a basic introduction to SCM and theories within AI. At the end of the chapter, related work that was an inspiration to our work is presented.

3.1 Supply Chain Management

In this section, some background theory about SCM is presented. It is mostly intended for those who have no former knowledge within this area.

3.1.1 The concept of SCM

There is a difference between logistics and SCM. Logistics is focused on optimizing individual parts of the supply chain, i.e., the own company. The more modern term SCM was first introduced by management consultants in the early 1980’s. SCM focuses on processes rather than functions, and these processes should be managed in such a way that maximum customer satisfaction is obtained (Weele 2014, p. 239). This means that rather than optimizing the individual companies, the entire supply chain should be optimized as a whole. The potential positive outcomes of SCM are widely spread, but there is a large gap between what theory suggests and how commercial companies actually work (Sweeney, Grant, and Mangan 2015).

According to KPMG (2016), a consultant agency, a new type of customer has emerged, which is more informed and demands a greater flexibility than before. The demands fluctuate and the requirements as well. At the same time, a global market with much competition makes it harder for companies to gain a competitive advantage. The customer wants the products delivered faster, damage-free and on time. If the product takes a long time to produce, the customer might choose another supplier, given a comparable quality. Companies today compete mostly on the basis of quality and time. This demands for closer collaboration with the suppliers, including information sharing (Mentzer et al. 2001).

In order to improve the supply chain, information sharing is needed. By sharing information and having an SCM approach, i.e., optimizing the whole supply chain
rather than a single entity, improved production planning, utilization of resources and customer service can be obtained, as well as reduced inventory costs and lead times (Kembro, Näslund, and Olhager 2017). Different obstacles make this information sharing difficult though. Information can be misinterpreted and incomplete, structures and culture can have an impact on a company’s willingness to share information (ibid.) and some companies fear that information will leak to competitors, and thereby reduce their competitiveness (M. Cooper, Lambert, and Pagh 1997). Previous studies have also shown that the IT systems the partners in the supply chain use are incompatible, making it difficult and costly to share information (Kembro, Näslund, and Olhager 2017). It can be difficult to share information when working in an advanced business system with a partner who uses a spreadsheet. These obstacles might be part of the reason why many companies are not working according to what theory suggests when it comes to SCM.

In the order handling area, companies should share demand data in order to improve planning and production. As the study by Kembro, Näslund, and Olhager (ibid.) shows, information that is delayed for some reason has nearly no value for the company. Therefore, quick information exchange is important, in order to be able to use this for decision making and making the company more competitive in an industry where the demand quickly changes.

3.1.2 The order process

There are three levels of purchasing; strategic, tactical and operative. The strategic level is about long-term decision making. The strategic purchaser’s role is to analyze the company’s spending and develop differentiated supplier strategies based on whether the suppliers are crucial for the business or not. Tactical purchasing is about standardizing purchasing processes, negotiating contracts with the suppliers and striving for improvement of the suppliers. This is based on the supplier strategies achieved in the first level. The last level is about the day-to-day decisions, processes and planning. The role of the the operational purchasers is to, at the lowest overall cost, secure supply at the right quality and quantity at the right time (Weele 2014, p. 3-4).

The process for the operational purchasing when doing traditional order placement in PipeChain can be seen in figure 3.1. The forecasts for the customer demand are generated and can be shared with suppliers when having long-term agreements. The supplier can then prepare for the future demand. An order can be placed exactly as what was forecasted, but can also deviate a lot since the real demand might differ from the forecasted one. An order usually consists of a couple of order lines, describing the quantity and demanded price for each of the components, i.e., each order line represents the order placement of a unique component. The order needs to be approved by the supplier in question, which is made through an order confirmation. It is in some cases possible to change or cancel the order without any repercussions before the order confirmation has been sent. The order is seen as finished once the invoice has been paid.
3.2 Artificial Intelligence

AI is a field within science and engineering where one tries to understand and build intelligent entities. There are different definitions of AI. One tends to let the different definitions concern how the entity is reasoning or behaving and whether this is close to either a human performance or an ideal performance. AI includes a large amount of sub-fields, including among other things learning, communicating and planning (Russell and Norvig 2016, p. 1-2).

Within AI there are different ways to handle uncertainty. Uncertainty can be due to non-determinism or partial observability. In the case of this project the uncertainty is due to the objective of trying to predict future events, i.e., non-determinism. One could use problem-solving agents or logical agents to handle uncertainty. These agents keep track of a belief state, which means they keep track of all possible states that they might be in and generate a plan for every possible state. These methods are however not considered to be suitable for the task at hand in this project. This is partly due to the nature of the task and partly due to the drawbacks that these approaches have, such as the fact that the agent must consider every possible explanation and plan for every state no matter how unlikely that state is, which can lead to very large amounts of calculations and stored data. The nature of the task at hand is predicting future events and the likelihood of them happening. In other words the task is to provide a degree of belief for a certain event to take place. A tool that can be used to handle these degrees of belief is probability theory. In probability theory an agent has a numerical degree of belief between zero and one for every state or sentence, meanwhile a logical agent only believes that a certain statement is either true or false. The probability statement is made with respect to the knowledge state, that is what the agent knows about the world, and not with respect to the actual world (ibid., p. 480-482). This is sometimes also referred to as probabilistic reasoning. Probabilistic reasoning is a sub-field within AI in which one builds network models under uncertainty according to the laws of probability theory (ibid., p. 510).
3.2.1 Absolute independence vs. conditional independence

In probabilistic reasoning it is important to differentiate between absolute independence and conditional independence. Two random variables $X_1$ and $X_2$ are absolutely independent if the result of $X_1$ in no way depends on the result of $X_2$ and vice versa (Russell and Norvig 2016, p. 494). Two random variables $X_1$ and $X_2$ are conditionally independent if the result of $X_1$ doesn’t depend on the result of $X_2$ and vice versa, given the result of a third variable $Z$ (ibid., p. 498-499). Consequently, a random variable $X_1$ is dependent on $X_2$ if the result of $X_1$ depends on the result of $X_2$.

In many real world situations one can end up with very large probabilistic domains as the number of variables grows, which can be a problem when trying to draw conclusions from the known data. Independencies can be an important base in decomposing these kind of large domains into smaller pieces, which in many cases simplify the calculations. The easiest way would be to totally divide the domain into smaller absolutely independent domains. Absolute independencies are rare compared to conditional independencies though. Therefore a great alternative is to use conditional independencies to separate large domains into smaller weakly connected domains (ibid., p. 499).

3.2.2 Bayesian networks

Bayesian networks constitute a systematic way to represent dependencies, absolute independencies and conditional independencies. The representation consists of a directed acyclic graph. Each node in the graph corresponds to a random variable and contains quantitative probability information about it. Each edge in the graph corresponds to a dependency. If there is an edge from node $X_1$ to node $X_2$, $X_1$ has a direct influence on $X_2$. Therefore each node $X_i$ has a conditional probability distribution $P(X_i|\text{Parents}(X_i))$ that quantifies the effect of the parents on the node. An example of a Bayesian network is shown in figure 3.2. In this example $Z$ has a direct influence on $X_1$ and $X_2$, while $X_1$ and $X_2$ are conditionally independent given the value of $Z$. $X_3$ is dependent on $X_2$ and $Y$ is absolutely independent on all other variables. In many cases the conditional distributions are also shown in so called conditional probability tables next to each node (ibid., p. 510-517).

Figure 3.2: Simple example of a Bayesian network.
Bayesian learning is a concept where an agent builds a Bayesian network that can be used to calculate probabilities for different hypotheses to be true. The agent builds the network by training on a so-called training dataset. The dataset contains instances which hold certain values for the predefined attributes. There are some main advantages of using Bayesian learning compared to other known techniques within AI. Firstly, one observed training example can increase or decrease the estimated probability for a certain hypothesis to happen, rather than completely excluding some hypotheses that are inconsistent with the single example. Secondly, one can combine former knowledge about the domain with observed data to find the probability of a certain hypothesis to be correct. Thirdly, Bayesian methods can make probabilistic predictions. This means an algorithm using Bayesian Networks can calculate the probabilities of different hypotheses rather than just predicting one likely hypothesis (Mitchell 1997, p. 154-155).

3.2.3 Learning algorithms that manipulate probabilities

In the previous section the structure of a Bayesian network was explained. There are several techniques that use Bayesian networks and probability theory to perform classifications and predictions. In this section some techniques that could be suitable for the prototype are presented.

Bayes optimal classifier

A classification of an instance is made by combining the predictions, weighted by their posterior probabilities, of all hypotheses. The probability that the instance is correctly classified can be calculated through the following formula:

\[
P(v_j|D) = \sum_{h_i \in H} P(v_j|h_i)P(h_i|D)
\]

where \(v_j\) is any classification value that the instance can take, \(P(v_j|D)\) is the probability that \(v_j\) is the correct classification given the dataset \(D\) and \(h_i\) is one possible hypothesis from the set \(H\). From this the optimal classification for an instance is the value \(v_j\), for which \(P(v_j|D)\) is maximal. This technique optimizes the probability of a new instance being correctly classified, but is costly to apply since one needs to compute the posterior probability of every hypothesis in \(H\) and then combine the results to make one classification (ibid., p. 175-176).
Naïve Bayes classifier

This technique is an alternative to Bayes optimal classifier. It reduces the cost of computation by assuming that all attribute values, given the target value, are conditionally independent. That means that the probability of observing the conjunction \(a_1, a_2...a_n\) of attribute values can be calculated as:

\[
P(a_1, a_2...a_n|v_j) = \prod_i P(a_i|v_j)
\]

where \(v_j\) is equivalent to the one used in Bayes optimal classifier (Mitchell 1997, p. 177). In this case the target value would be the classification itself and the attribute values could for example be the ordered item, the order date, the wanted delivery date and the order quantity. The assumption that all attribute values are conditionally independent given the target value can be represented by the Bayesian network in figure 3.3, where \(Z\) is the target value and \(X_1,...,X_n\) are the attribute values.

3.2.4 Decision trees

Decision tree induction is a great form of machine learning and is also very simple. The learning algorithm builds a so called decision tree from the training data and makes decisions based on it. The algorithm takes test data with different attribute values as input and outputs a single value, the classification. Hence, it returns a form of decision. In order to reach this decision, it performs a series of tests using the decision tree. Every node in the tree represents a test of the value of one attribute and the leaf nodes establish a value that should be returned. The algorithm works very well under some circumstances, but unfortunately not in all situations (Russell and Norvig 2016, p. 697-699). An example of a decision tree is shown in figure 3.4. The figure illustrates a decision problem with only two choices; yes or no. The variables \(X_1, X_2\) and \(X_3\) represent attributes and each arrow represent a specific value of the parent attribute.
3.2.5 Validation

When training a model, it is important to consider how to do so in order to get the best result. There should be enough instances to train on, in order to give a satisfying result when testing. Overfitting the model must be also avoided. Overfitting is when the model overreacts to minor changes in the test data, which makes the model specialized to the training data and thereby unfit when it comes to new test data (ibid., p. 805).

There are two common validation schemes, $k$-fold cross-validation and hold-out validation (Yadav and Shukla 2016). The former makes the data set serve as both training and test data. After splitting the data into $k$ subsets, $k$ rounds of learning are conducted. In each round, $1/k$ of the data serves as test data and the rest as training data. The average test score should then be a better estimate compared to just conducting a single score (Russell and Norvig 2016, p. 708). The accuracy in $k$-fold cross-validation increases as $k$ increases, but this also raises the risk of overfitting (Yadav and Shukla 2016). Hold-out validation splits the data set randomly into a training set and a test set where the accuracy is evaluated. This is only done once. The disadvantage of this method is that it is impossible to find the perfect split between test data and training data. If having a large amount of test data, there are less data to train on and vice versa (Russell and Norvig 2016, p. 708).

The generally, suggested approach is to use hold-out validation when the data set is huge, since the $k$-fold cross-validation will require a great amount of computational time for large data sets (Yadav and Shukla 2016).
3.3 Related work

A study by Dogan and Aydin (2011) investigates how Bayesian networks and Total Cost of Ownership can be combined to analyze supplier selection. The researchers analyze former data and interview experts to find a model that can be used when a customer is to choose a supplier of a certain product. The topic is similar to the one studied in this project, though the focus is on strategic purchasing. The main difference between the studies is what data can be analyzed and the recurrence of the analyzed events. The placing of an order is in most cases done much more often than the selection of a supplier. The researchers highlight the main reasons to why they chose to use Bayesian networks. Some of the reasons were:

- Bayesian networks are powerful in drawing conclusions from known data.
- Bayesian networks can model uncertainty by allowing probability distributions.
- One can combine expert knowledge with correlations found in the data.

All these advantages closely correspond to the desired qualities of the model used in this project. This strengthens our belief that Bayesian networks are a good starting point for the project. The researchers also pinpoint that first-time supplier selections cause difficulties in the analysis since there is not as much data to base the decision on. Nevertheless, they reckoned that their Bayesian network was good at handling this kind of uncertainty. In our project there will also be cases where there is not a lot of former data to analyze and a model that can handle uncertainty is needed.

In a study by Ahn and Ezawa (1997), Bayesian networks were used to provide telemarketers with decision support on when they should invest time in pitching discount plans to the customer they have on the line. The decision support was aimed at minimizing the costs of spending time talking to a customer that would reject the offer. They used the Bayesian network to classify customers as either "takers" or "nontakers" of new offers based on previous response history. There were 300 variables describing customer behavior to choose from. By using expert knowledge these were narrowed down to 30 variables that the learning algorithm was tested on. These 30 variables were later on narrowed down to 12 variables by using entropy measures. An important part of our project will be to choose the right variables as input to the learning algorithm and the methods used in this study could help in doing so.

According to Pavón et al. (2011) a dataset where some classes are vastly more represented than others is a well-known problem within machine learning, which can decrease the performance of many classifiers. The classifier might then have problems with classifying the classes that are less represented. This could be considered to be irrelevant when using Bayesian networks which are based on probabilities, since the uneven distribution represents the real-life condition. But as Pavón et al. (ibid.) argues, also Bayesian networks can perform better if this unbalance is handled. If the data analyzed in this project is imbalanced, methods to handle this will therefore be investigated.
4 | Applying AI to order placement

One part of this project has been to implement a prototype that analyzes customer
data and predicts future events. This chapter describes the tool that has been used
and how the testing of different settings and attributes was conducted.

4.1 Weka

Weka is an open source software tool developed by scientists at the university of
Waikato in New Zealand. The tool offers a mixture of machine learning algorithms,
including Bayesian networks and decision trees, that can be used to analyze and
classify data. Weka includes a graphical user interface, GUI, where users can upload
their own data and apply the existing algorithms to it. It also includes an API which
enables usage of the algorithms from Java code projects. Weka can among other
things be used to pre-process, classify and visualize datasets. It provides various
filters that can filter the data from certain instances or attributes. Weka’s tool
bench also includes algorithms for selecting attributes to include in the analysis.
The system can be used to compare different learning algorithms to each other
(Weka 2017).

We have used Weka in order to test which attributes that are suitable, what learning
algorithm to use and which settings that give the best performance of the algorithm.
This has made the experimental part of the project much smoother, since there was
no need to program every test. When an algorithm and all its settings had been
chosen, based on the experimental testing, Weka’s API was used to program the
prototype.

4.1.1 Algorithms

As mentioned above Weka includes a large amount of different learning algorithms.
To get an overview on which algorithms that could eventually perform well on the
given dataset, several initial tests were performed. The decision on which
algorithms to test further was based on the theoretical background knowledge
about the algorithms that is presented in section 3.2, what algorithms that were
possible to use on the dataset at hand and which algorithms performed the best.
The algorithms that we chose to investigate further are presented in this section. The results from the further testing on these algorithms are shown in section 4.3.2.

ZeroR

ZeroR is a baseline algorithm that ignores the attributes and simply looks for the most common classification in the training dataset. It then classifies all test instances as belonging to this classification. If ZeroR has a good performance it means that the data is not evenly distributed over all classifications. The distribution of the classifications has a great impact on the performance of other algorithms. Therefore ZeroR gives a baseline performance that we always want to compare the performance of other algorithms to. An algorithm that performs worse than ZeroR is useless. This comparison can also explain a part of the reason why the performance might differ a lot when running the same algorithm on different datasets (Frank, Hall, and Witten 2017).

BayesNet

The BayesNet algorithm is Bayes optimal classifier. If the data set contains continuous variables, the algorithm discretizes them before starting the analysis. It has a default way of handling missing values in the dataset. For numeric attributes it replaces the missing value with the mean of the instances in the dataset. For nominal attributes, it replaces missing values in the instance with the most common value in the rest of the dataset.

Learning a Bayesian network is a process in two steps: the first is to learn the network structure and the second to learn the probability tables. The Bayesian learner uses a search algorithm to search through the space of potential network structures. There are different search algorithms and the following can be chosen in Weka:

- **K2**: Starts by presuming that a node has no parents. Using greedy search, it then adds the parent whose contribution increases the score of the structure the most. It stops when adding parents to the node no longer increases the score. (Lerner and Malka 2011)

- **Hill climbing**: A loop that moves in the direction where the neighbour has a higher value. When no immediate neighbor has a higher value, it stops (Russell and Norvig 2016, p. 122).

- **Simulated annealing**: As hill-climbing does not allow downhill moves, a random move in any way is added in this search algorithm, to avoid not getting stuck in a local maximum. Given that the move improves the situation, it is accepted and if not, it is accepted with a probability less than one (ibid., p. 125).
4.1. WEKA

- **Repeated hill climber:** The starting point is a randomly generated network. The Hill climber is used to reach a local optimum (Bouckaert 2008).

- **LAGD Hill Climbing:** A set of best scoring steps are looked ahead at, while performing Hill climbing (ibid.).

- **TAN, Tree Augmented Naïve Bayes:** The structure of the tree is made by computing the maximum weight spanning tree, as discussed by Dorrigiv et al. (2015).

- **Tabu search:** A variant of Hill climbing. When a local optimum is found, it steps to the neighbourhood candidate that is least bad. It knows which is worse since it keeps a list of the $k$ states that cannot be revisited (Russell and Norvig 2016, p. 154).

- **Genetic search:** Uses a genetic search algorithm, i.e., a search that replicates natural selection. (Bouckaert 2008).

There are some options that these search algorithms have in common. These are:

- **initAsNaïveBayes**
  - When true, which it is by default, the initial network structure is based on the Naïve Bayes assumption.
  - When false, a network structure without any arrows is used initially.

- **markovBlanketClassifier:**
  - When true, a heuristic in the end of the traversal of the search space makes sure that all attributes are in the classifier node’s Markov blanket, i.e, each node is a sibling, parent or child to the classifier node.
  - When false, which it is by default, an arrow will be added.

- **scoreType:** Determines the score type used. The score type is used to evaluate a network structure in order to be able to compare it to other structures.

- **maxNbrOfParents:** Gives an upper bound on the number of parents of each node. This is set to one by default.

**NaïveBayes**

This algorithm is based on Bayes’ rule and the Naïve Bayes assumption that all attributes are conditionally independent of each other. Missing values are no problem for this algorithm. If there is a missing value in the training data, the probability ratios are based on the values that actually occur rather than all instances (Frank, Hall, and Witten 2017, p. 99-100). It assumes that all numeric attributes are normally distributed, but it is possible to choose other distributions if knowing which one is correct (ibid., p. 105).
Decision tree - J48

In Weka, a decision tree algorithm called J48 is available. A rule is generated for each leaf and makes a connection between all tests from the root to the leaf. In order to prune the tree, once a new rule is formulated, each condition is tested by removing it and computing which of the training examples that are covered by the new rule. From this, a pessimistic error rate estimate is made both on the new rule and the old rule. If the new one is better, the condition should be removed. The algorithm continues doing this until there are no more rules to delete in order to improve (Frank, Hall, and Witten 2017, p. 219).

AttributeSelectedClassifier

This classifier divides the classification problem into two steps. During the first step it tries to reduce the dimension of the problem by selecting a subset of the available attributes to be part of the analysis. During the second step a machine learning algorithm from Weka’s library, for example BayesNet or J48, is used to build a learning model based on only the selected subset of attributes. The reduction of attributes can improve the performance of the learning algorithm significantly, since it removes attributes that otherwise might have confused the algorithm.

In general, one wants to include the attributes that contribute the most to a good performance. There are many different evaluation methods that can be used to find these attributes. When using the AttributeSelectedClassifier one can choose from seven different evaluators. Some of the evaluators rank the attributes according to the contribution to a good performance. This ranking might be based on, for example, the gain ratio or the info gain of the attributes. Info gain is biased toward attributes having many values while the gain ratio, which is an extension to info gain, overcomes this bias. In some way it normalizes the info gain, by dividing the gain score with the split information, i.e., the entropy of the the test outcome (Harris 1983).

\[
\text{Gain ratio} = \frac{\text{Info gain}}{\text{Split information}}
\]

The ranking evaluators are very fast, but they do not make allowances for the interaction between different attributes. This means that the attributes that have been top ranked might not be the best combination of attributes to use. Therefore the ranking evaluators will not be used in this project. To find the best combination of attributes one needs to search through the space of attribute subsets and evaluate each subset. Out of the seven evaluators available in Weka, there are two evaluators that evaluate subsets of attributes instead of evaluating a single attribute at a time:

- **CfsSubsetEvaluator**: Evaluates the value of a subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them. This means that the evaluator will give lower scores to the subsets where some of the attributes are correlated to each other.
4.2. DATA CLEANSING

• WrapperSubsetEvaluator: Evaluates attribute sets by using a learning scheme. This means that the evaluator itself uses one of Weka’s learning algorithms to try to find the best subset of attributes. It is recommended to use the same algorithm for this evaluation as the one being used for the second step mentioned above.

As mentioned above one needs to search through the space of possible attribute subsets to find subsets to evaluate. There are two different search methods available for the AttributeSelectedClassifier, GreedyStepwise and BestFirst. Since many greedy algorithms do not give an optimal solution (Kleinberg and Tardos 2014, p. 157-184), only BestFirst was used.

The chosen evaluator and search method are used in the first step of the AttributeSelectedClassifier to find a subset of attributes that are likely to contribute to a good performance. When a subset of attributes has been found the AttributeSelectedClassifier goes on to the next step, which simply involves running a selected learning algorithm on the data set with only the selected attribute subset (Weka 2017).

4.2 Data cleansing

When looking at the data we found that some instances included values that might be somewhat confusing to the algorithm. Therefore some data cleansing was done before starting the analysis of the data. The data was cleansed based on the following properties:

• Instances where the order date was after the demanded delivery date were found in the database. According to the interviewed consultants, the purchasers seem to think they will be prioritized by the supplier when doing so. This means that the demanded delivery date is in the past when the order is being placed, which will automatically make the order late. This is not the situation where the algorithms we are examining will be used. Such an order would have a 100% probability of arriving late, which makes this kind of algorithm useless. Therefore these instances were removed.

• Order lines can have different statuses. We chose to only look at order lines with the status finished, since this means that everything has been delivered and invoiced.

• In some cases customers have changed the order line after the order date. We chose to only look at order lines that had not been changed, since it is hard to predict how changes affect the outcome.
4.3 Attribute selection

One of the challenges of building a learning algorithm that performs well is deciding on which attributes to consider in the analysis. Having attributes which are irrelevant or redundant can cause the performance of the algorithms to deteriorate (Frank, Hall, and Witten 2017, p. 287). This section describes how we have conducted the attribute selection in order to improve the performance of the tested algorithms.

4.3.1 Base attributes

Since PipeChain has access to databases with approximately 160 attributes, all of them cannot possibly be part of the analysis. Inspired by Ahn and Ezawa (1997), a first selection was made based on knowledge within the SCM field and the answers we got from interviews with consultants and customers. The following attributes were considered important and were regarded during experimental testing:

- **ProdID**: This attribute represents the product that is being ordered. The order and delivery patterns might differ a lot between different products.

- **DmdPrice**: This attribute represents the unit price of the ordered product. Interviews with the customers suggest that price often indicates the level of complexity of the product. For a more complex product, the delivery time is often longer.

- **DmdQty**: This attribute represents the demanded quantity of the product that is being ordered. Larger quantities might lead to longer delivery times. Some quantities can also be a better match to the suppliers’ batch sizes, making it easier for the suppliers to deliver.

- **OrderLinePrice**: This attribute represents the total price of one order line, hence it is given by:

  \[ DmdPrice \times DmdQty \]

  Price and quantity are often correlated. Expensive products can often be ordered in small amounts and cheap products can often be ordered in large amounts. Therefore only looking at DmdQty might be misleading. Looking at the multiplied value of price and quantity might make the algorithm see when an order line puts unusually large pressure on the supplier, which might make it harder to deliver on time.

- **OrderMonth**: This attribute represents the month of the order date and makes the algorithm look for patterns of seasonality in the dataset. According to some interviewees the delivery order pattern might differ between different parts of the year. For example the summer months might be calmer due to vacations.
4.3. ATTRIBUTE SELECTION

- **OrderWeekday**: This attribute represents the weekday of the order date and makes the algorithm look for weekly patterns in the dataset. According to some interviewees the delivery and ordering pattern might differ between different days of the week. Consider a supplier who sets his production schedule once a week. If the order arrives after the planning has taken place at the supplier, they could have a difficult time to deliver on the demanded date.

- **NbrOfDays**: This attribute represents the number of days between the order date and the demanded delivery date. The larger number of days the more time the supplier has to produce and deliver the order. At the same time, there is a higher risk of the order changing when it is placed very early. Therefore this number should be highly correlated with the ability to deliver on time.

4.3.2 Experimental testing

Experimental testing was performed in order to see which of the base attributes that contribute to better performance of the algorithms. We started by only considering the data from one customer and trying to find an algorithm and a selection of attributes that gave good predictions based on this data. We started by looking at one customer-supplier relationship, and not on all suppliers at once. This was mainly in order for the algorithm to easier detect patterns with a minor set of data.

**Test customer**

The customer had 269 722 instances in the database. After doing the data cleansing mentioned in section 4.2, the number of instances was reduced to 227 207. Most of the orders were placed during 2017. During this time interval, the customer had placed orders with 381 unique suppliers.

**Test suppliers**

We looked at two different customer-supplier relationships. The choice of suppliers was based on how many orders the suppliers had in the database of the test customer. We wanted as much data as possible, therefore the two suppliers with the highest amount of order lines were chosen.

*Supplier 1*

There were 13 985 order lines with this supplier after doing the data cleansing mentioned in section 4.2. As mentioned in section 4.1.1 the distribution of the order classifications has a great impact on the performance of the algorithms on the data set. The classification distribution of the dataset is shown in figure 4.1. Due to this distribution the algorithm ZeroR has a quite high accuracy. When running it on this dataset with 10-fold cross-validation it had an accuracy with a mean of 56.94% and a standard deviation of 0.09.
CHAPTER 4. APPLYING AI TO ORDER PLACEMENT

There were 3985 order lines with this supplier after doing the data cleansing. The classification distribution of the dataset is shown in figure 4.2. Due to this distribution, when running the ZeroR algorithm on this dataset with 10-fold cross-validation it had a quite low accuracy with a mean of 28.95% and a standard deviation of 0.12.

Figure 4.1: The distribution of the order line classes in the dataset of Supplier 1.

Supplier 2
There were 3985 order lines with this supplier after doing the data cleansing. The classification distribution of the dataset is shown in figure 4.2. Due to this distribution, when running the ZeroR algorithm on this dataset with 10-fold cross-validation it had a quite low accuracy with a mean of 28.95% and a standard deviation of 0.12.

Figure 4.2: The distribution of the order line classes in the dataset of Supplier 2.

Manual attribute selection

We started the experimental testing by trying to manually find the attributes that gave the best performance. We did this by starting with two of the base attributes and then adding one attribute at a time to see which attributes improved the performance. The two starting attributes were the ones that we thought would have the greatest importance, DmdQty and NbrOfDays. These were also the attributes that seemed most reasonable to start with according to the interviewees.

During testing we used 10-fold cross-validation and repeated each round 10 times in order to receive the mean and standard deviation of the classification accuracy. Standard deviation can be seen as the algorithm’s tendency to learn random things despite what the real signal is saying, and can thereby be seen as a measure of overfitting (Domingos 2012).
4.3. ATTRIBUTE SELECTION

The accuracy is measured as the share of correctly classified instances and is therefore given in percent. When comparing different algorithms, NaïveBayes and J48 were compared to BayesNet. When an algorithm is significantly better than BayesNet we display this by adding a star (*) next to the mean and when an algorithm is significantly worse we mark this by adding a single quotation mark (’) next to the mean. A performance is considered significantly better or worse if the performance level differs at the specified significance level of 5%. The first row in the tables show the performance of the initial round. The following rows show the performance of the subsequent rounds. The column called "Configuration" shows what attribute has been added compared to the initial round. If the result of a subsequent round is significantly better than the result of the initial round, the mean is marked in bold and if the result of a subsequent round is significantly worse, the mean is marked in italic. The result from this manual testing on the dataset of Supplier 1 and Supplier 2 can be seen in table 4.1 and table 4.2.

Table 4.1: The mean and standard deviation of the accuracy recorded during the manual test rounds with 5000 instances from the dataset of Supplier 1.

<table>
<thead>
<tr>
<th>Round</th>
<th>Configuration</th>
<th>BayesNet</th>
<th>Naïve Bayes</th>
<th>J48</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Mean</td>
<td>Std.d</td>
<td>Mean</td>
</tr>
<tr>
<td>Initial</td>
<td>Orig. attr.</td>
<td>82.34</td>
<td>138</td>
<td>73.76</td>
</tr>
<tr>
<td>Round 1</td>
<td>+ProdId</td>
<td>80.97</td>
<td>135</td>
<td>72.92</td>
</tr>
<tr>
<td>Round 2</td>
<td>+OrderLinePrice</td>
<td>82.71</td>
<td>136</td>
<td>76.52</td>
</tr>
<tr>
<td>Round 3</td>
<td>+OrderMonth</td>
<td>83.21</td>
<td>1.33</td>
<td>62.94</td>
</tr>
<tr>
<td>Round 4</td>
<td>+Price</td>
<td>81.79</td>
<td>1.29</td>
<td>63.37</td>
</tr>
<tr>
<td>Round 5</td>
<td>+OrderWeekday</td>
<td>82.31</td>
<td>1.48</td>
<td>72.40</td>
</tr>
</tbody>
</table>

Table 4.2: The mean and standard deviation of the accuracy recorded during the manual test rounds with all the instances from the data set of Supplier 2.

<table>
<thead>
<tr>
<th>Round</th>
<th>Configuration</th>
<th>BayesNet</th>
<th>Naïve Bayes</th>
<th>J48</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Mean</td>
<td>Std.d</td>
<td>Mean</td>
</tr>
<tr>
<td>Initial</td>
<td>Orig. attr.</td>
<td>53.78</td>
<td>189</td>
<td>33.54</td>
</tr>
<tr>
<td>Round 1</td>
<td>+ProdId</td>
<td>55.33</td>
<td>206</td>
<td>40.67</td>
</tr>
<tr>
<td>Round 2</td>
<td>+OrderLinePrice</td>
<td>58.10</td>
<td>208</td>
<td>35.37</td>
</tr>
<tr>
<td>Round 3</td>
<td>+OrderMonth</td>
<td>58.85</td>
<td>208</td>
<td>38.51</td>
</tr>
<tr>
<td>Round 4</td>
<td>+Price</td>
<td>58.00</td>
<td>199</td>
<td>34.39</td>
</tr>
<tr>
<td>Round 5</td>
<td>+OrderWeekday</td>
<td>53.82</td>
<td>198</td>
<td>33.68</td>
</tr>
</tbody>
</table>

All algorithms performed significantly better than ZeroR in every round on the dataset of Supplier 1. This indicates that the tested algorithms do find some patterns.
in the data. NaïveBayes performed *significantly* worse than BayesNet in all of the rounds. J48 performed *significantly* better than BayesNet in the majority of the rounds. We can see that OrderLinePrice and OrderMonth improved the BayesNet algorithm and that OrderMonth and OrderWeekday improved the J48 algorithm.

All algorithms performed *significantly* better than ZeroR in every round on the dataset of Supplier 2. NaïveBayes performed *significantly* worse than BayesNet in all of the rounds. J48 performed *significantly* better than BayesNet in the majority of the rounds. We can see that all attributes except for OrderWeekday improved the BayesNet algorithm, that ProdId, OrderLinePrice and OrderMonth improved the NaïveBayes algorithm and that all attributes except for ProdId improved the J48 algorithm.

It can be seen that all algorithms perform worse on the dataset of Supplier 2 than on the dataset of Supplier 1. This is partly due to the fact that the dataset of Supplier 2 is very evenly distributed, which makes it harder for the algorithm to make a good guess of the outcome. It could also be an indication that the behavior of Supplier 2 is more inconsistent and therefore harder to predict.

To see if the combination of the **bold** attributes would improve the performance further we added them in different combinations and compared them to some of the former results. We only did this for Supplier 1. The results can be seen in table 4.3, 4.4 and 4.5. A star or single quotation mark still means that the algorithm performed *significantly* better or worse than BayesNet. **Bold** or *italic* text means that the combination of attributes that was used in the round was *significantly* better or worse, respectively, than the single attribute that can be seen in the first row.

*Table 4.3: The mean and standard deviation of the accuracy recorded when adding the combination OrderLinePrice and OrderMonth in comparison to adding only OrderLinePrice with 5000 instances from the dataset of Supplier 1.*

<table>
<thead>
<tr>
<th>Round</th>
<th>Configuration</th>
<th>BayesNet</th>
<th>Naïve Bayes</th>
<th>J48</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Mean</td>
<td>Std.d</td>
<td>Mean</td>
</tr>
<tr>
<td>Round 2</td>
<td>+OrderLinePrice</td>
<td>82.71</td>
<td>136</td>
<td>76.52</td>
</tr>
<tr>
<td>Round 7</td>
<td>+OrderLinePrice + OrderMonth</td>
<td>83.21</td>
<td>146</td>
<td>67.44</td>
</tr>
</tbody>
</table>
4.3. ATTRIBUTE SELECTION

Table 4.4: The mean and standard deviation of the accuracy recorded when adding the combination of OrderLinePrice and OrderMonth and the combination of OrderMonth and OrderWeekday in comparison to adding only Month with 5000 instances from the dataset of Supplier 1.

<table>
<thead>
<tr>
<th>Round</th>
<th>Configuration</th>
<th>BayesNet Mean</th>
<th>BayesNet Std.d</th>
<th>NaïveBayes Mean</th>
<th>NaïveBayes Std.d</th>
<th>J48 Mean</th>
<th>J48 Std.d</th>
</tr>
</thead>
<tbody>
<tr>
<td>Round 3</td>
<td>+ OrderMonth</td>
<td>83.21</td>
<td>13</td>
<td>62.94</td>
<td>200</td>
<td>85.60 *</td>
<td>120</td>
</tr>
<tr>
<td>Round 7</td>
<td>+ OrderLinePrice</td>
<td>83.21</td>
<td>14</td>
<td>67.44</td>
<td>243</td>
<td>85.43 *</td>
<td>116</td>
</tr>
<tr>
<td>Round 8</td>
<td>+ OrderWeekday</td>
<td>83.35</td>
<td>13</td>
<td>63.20</td>
<td>209</td>
<td>87.67 *</td>
<td>122</td>
</tr>
</tbody>
</table>

Table 4.5: The mean and standard deviation of the accuracy recorded when adding the combination of OrderWeekday and OrderMonth in comparison to adding only OrderWeekday with 5000 instances from the dataset of Supplier 1.

<table>
<thead>
<tr>
<th>Round</th>
<th>Configuration</th>
<th>BayesNet Mean</th>
<th>BayesNet Std.d</th>
<th>NaïveBayes Mean</th>
<th>NaïveBayes Std.d</th>
<th>J48 Mean</th>
<th>J48 Std.d</th>
</tr>
</thead>
<tbody>
<tr>
<td>Round 6</td>
<td>+ OrderWeekday</td>
<td>82.31</td>
<td>14</td>
<td>72.40</td>
<td>506</td>
<td>84.64 *</td>
<td>131</td>
</tr>
<tr>
<td>Round 8</td>
<td>+ OrderWeekday</td>
<td>83.35</td>
<td>13</td>
<td>63.20</td>
<td>209</td>
<td>87.67 *</td>
<td>122</td>
</tr>
</tbody>
</table>

We can see that the only combination of attributes that performed significantly better in comparison to only adding one of the attributes was the combination of OrderWeekday and OrderMonth when using J48. Moreover it is hard to draw any conclusions from the recorded results. We therefore decided to go on with automatic attribute selection.

Automatic attribute selection

In this section the importance of the attributes is evaluated using Weka’s tool for scheme-dependent attribute selection. The method tries to find the best subset of attributes by searching through the space of attribute subsets and evaluating each subset. This is done in the same way as when running the first step of the AttributeSelectedClassifier described in section 4.1.1. We used the WrapperSubset-Evaluator with BayesNet, NaïveBayes and J48 to evaluate each subset and the BestFirst search method to search through the space of subsets. The method uses 10-fold cross validation and shows the number of times the attributes have been part of the selected subset (Frank, Hall, and Witten 2016, p. 45). By using this
tool with forward, backward and bi-directional BestFirst search and combining the results of these 30 rounds, the results shown in table 4.6 and 4.7 were achieved for the different algorithms.

Table 4.6: The result from evaluating the attributes with the different algorithms on 5000 instances from the dataset of Supplier 1, using 10-fold cross-validation. The numbers show the share of the times in which the attribute was a part of the selected subset.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>BayesNet</th>
<th>Naïve Bayes</th>
<th>J48</th>
</tr>
</thead>
<tbody>
<tr>
<td>DmdPrice</td>
<td>6,7 %</td>
<td>0 %</td>
<td>30 %</td>
</tr>
<tr>
<td>DmdQty</td>
<td>60 %</td>
<td>0 %</td>
<td>100 %</td>
</tr>
<tr>
<td>ProdId</td>
<td>0 %</td>
<td>0 %</td>
<td>0 %</td>
</tr>
<tr>
<td>OrderMonth</td>
<td>100 %</td>
<td>0 %</td>
<td>100 %</td>
</tr>
<tr>
<td>NbrOfDays</td>
<td>100%</td>
<td>100 %</td>
<td>100 %</td>
</tr>
<tr>
<td>OrderLinePrice</td>
<td>46,7 %</td>
<td>0 %</td>
<td>30 %</td>
</tr>
<tr>
<td>OrderWeekday</td>
<td>76,7%</td>
<td>30 %</td>
<td>100 %</td>
</tr>
</tbody>
</table>

For the dataset of Supplier 1 only BayesNet had different values for forward, backward and bi-directional search, the other two algorithms had the same percentage regardless of which search method that was used. It can be seen that NbrOfDays was part of the selected subset in 100% of the cases for all algorithms, which supports our belief that NbrOfDays is an important attribute. It can also be seen that ProdId was not part of any of the selected subsets for any of the algorithms, which indicates that it is not necessary to know which product is being ordered to make a good prediction when looking at this supplier.

Table 4.7: The result from evaluating the attributes with the different algorithms on all instances from the dataset of Supplier 2, using 10-fold cross-validation. The numbers show the share of the times in which the attribute was a part of the selected subset.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>BayesNet</th>
<th>Naïve Bayes</th>
<th>J48</th>
</tr>
</thead>
<tbody>
<tr>
<td>DmdPrice</td>
<td>43,3 %</td>
<td>0 %</td>
<td>100 %</td>
</tr>
<tr>
<td>DmdQty</td>
<td>100 %</td>
<td>0 %</td>
<td>80 %</td>
</tr>
<tr>
<td>ProdId</td>
<td>100 %</td>
<td>100 %</td>
<td>0 %</td>
</tr>
<tr>
<td>OrderMonth</td>
<td>100 %</td>
<td>100 %</td>
<td>100 %</td>
</tr>
<tr>
<td>NbrOfDays</td>
<td>100 %</td>
<td>0 %</td>
<td>100 %</td>
</tr>
<tr>
<td>OrderLinePrice</td>
<td>56,7 %</td>
<td>0 %</td>
<td>10 %</td>
</tr>
<tr>
<td>OrderWeekday</td>
<td>90 %</td>
<td>90 %</td>
<td>100 %</td>
</tr>
</tbody>
</table>

For the dataset of Supplier 2 only BayesNet had different values for forward, backward and bi-directional search, the other two algorithms had the same percentage regardless of which search method that was used. It can be seen that
OrderMonth was part of the selected subset in 100% of the cases for all algorithms, which indicates that there are seasonal variations affecting Supplier 2. It can also be seen that there was no attribute that was not part of any of the selected subsets for any of the algorithms, which indicates that all of the attributes are of some importance when predicting based on this dataset. It could also be an indication that this supplier is more unpredictable than Supplier 1.

It is interesting to see that the subset of attributes selected when using the dataset from Supplier 1 differs from the subset of attributes selected when using the dataset from Supplier 2. This means that different suppliers might have different behavioral patterns and that it might be good to use different models for different customer-supplier relationships.

After running the attribute selector we ran the AttributeSelectedClassifier in order to classify based on only the selected attributes. Therefore the numbers in table 4.6 and 4.7 closely describe which attributes that were selected by this algorithm for the different suppliers. A comparison between the two evaluators available for the AttributeSelectedClassifier was done. Since the WrapperSubsetEvaluator was constantly better than the CfsSubsetEvaluator, it was used in all documented tests using the AttributeSelectedClassifier. The results of running the AttributeSelectedClassifier with BayesNet, NaïveBayes and J48 on the datasets from Supplier 1 and Supplier 2 are shown in table 4.8. We ran the algorithm on the dataset of Supplier 1 twice, once with 5000 instances and once with 10000 instances, to see if the result differed a lot depending on the amount of data.

Table 4.8: The results from running the AttributeSelectedClassifier with BayesNet, NaïveBayes and J48 with 10-fold cross-validation.

<table>
<thead>
<tr>
<th>Supplier</th>
<th>#Instances</th>
<th>BayesNet Mean</th>
<th>BayesNet Std.d</th>
<th>NaïveBayes Mean</th>
<th>NaïveBayes Std.d</th>
<th>J48 Mean</th>
<th>J48 Std.d</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supplier 1</td>
<td>5000</td>
<td>83.34</td>
<td>1.42</td>
<td>77.38</td>
<td>2.36</td>
<td>87.59</td>
<td>1.28</td>
</tr>
<tr>
<td>Supplier 1</td>
<td>10000</td>
<td>83.66</td>
<td>1.00</td>
<td>78.16</td>
<td>1.04</td>
<td>88.01</td>
<td>0.81</td>
</tr>
<tr>
<td>Supplier 2</td>
<td>All</td>
<td>61.68</td>
<td>2.03</td>
<td>50.94</td>
<td>2.35</td>
<td>69.47</td>
<td>2.03</td>
</tr>
</tbody>
</table>

When looking at the results of running the AttributeSelectedClassifier on the dataset of both Supplier 1 and Supplier 2 it can be seen that the algorithms sometimes perform better with the automatic attribute selection. Above all, the results are not significantly worse than when using manual attribute selection. This means a lot of manual work can be saved by automating the attribute selection process entirely, while getting at least as good a result as when doing it by hand.
4.4 Choice of algorithm

When choosing between different algorithms it is important to consider how much data is needed to achieve a good performance. To see how well the Attribute-SelectedClassifier performs on different amounts of data we used learning curves, which can be seen in figure 4.3 and 4.4. The learning curves were produced by running the algorithm on different numbers of training instances and plotting it against the accuracy of the algorithm. Normally the curve flattens out when reaching some number of training instances. This indicates that one should use at least this number of instances to get a good performance.

![Learning Curves](image)

Figure 4.3: The learning curves achieved from running the AttributeSelectedClassifier with BayesNet, NaïveBayes and J48 on the dataset of Supplier 1.

As can be seen in figure 4.3, J48 performs very well on this dataset. BayesNet’s performance is very stable, regardless of the number of instances used, while NaïveBayes reaches a stable performance at the end of the graph. Based on these learning curves, one would presumably choose to use J48 when having access to a larger dataset. But there is one great disadvantage of using J48. We want to retrieve a probability for the predicted outcome from our algorithm, which is very easy when using Bayesian learners and complicated when using decision trees. One can calculate the probability from a decision tree like J48 by calculating the frequency of each class in a leaf, but since J48 splits the training data into smaller and smaller pieces, this could be time consuming and give less reliable estimates than a Bayesian network (Frank, Hall, and Witten 2017, p. 340).
The learning curves achieved from running the AttributeSelectedClassifier with BayesNet, NaiveBayes and J48 on the data set of Supplier 2.

The standard deviations for the curves in figure 4.4 were very big, especially for J48. This, again, indicates that the patterns in the dataset of Supplier 2 are not as distinct as in the dataset of Supplier 1. The performance of J48 varies a lot with the number of instances, and does not seem reliable on this dataset.

In conclusion, when considering the stability, the accuracy, the possibility to get a probability for the predicted outcome and the computational power needed from the algorithm, BayesNet was considered the best choice for the prototype.

4.5 Refining the chosen algorithm

After choosing which algorithm to proceed with, refinement of this algorithm started. This was done by testing different settings in Weka. By setting initAsNaiveBayes to false, an initial network with no arrows was used. This was combined with changing the maximal number of parents for each node from one to three. This gave an accuracy of 86.55% for Supplier 1 with 10,000 instances and 66.55% for Supplier 2 with all instances, which both are far better results than previously achieved. The network structures are shown in figure 4.5. For all the earlier runs, where initAsNaive was set to true, the final network structure was Naive.
CHAPTER 4. APPLYING AI TO ORDER PLACEMENT

Figure 4.5: The network structures achieved when using an initial network with no arrows and a maximal number of parents set to three. The graph to the left was achieved when using 10,000 instances from the dataset of Supplier 1. The graph to the right was achieved when using all of the instances from the dataset of Supplier 2.

The challenge of finding a good algorithm does not only mean achieving a good accuracy. It is also important to look at what the algorithm predicts when it’s wrong and which classifications it has a harder time to predict. This can be done by viewing a so called confusion matrix. The confusion matrix shows what the instances were classified as by the algorithm in comparison to the real class of the instances. This is interesting to view since some misclassifications are more severe than others. In table 4.9 and 4.10, the confusion matrices for when setting initNaïveBayes to true respective false are shown. The values are also shown in bar charts to visualize the differences between the two settings in figure 4.6 and 4.7. We ran the AttributeSelectedClassifier on 10,000 instances from the dataset of Supplier 1. The headlines (">2 days early", "on time" etc.) show what class the algorithm classified the instances as. Each bar chart shows the distribution of the real outcomes for all instances that the algorithm classified as the class written in the headline. Each bar shows the share of instances that had the real outcome stated below the bar. The stripes highlights the correct classifications and therefore the striped bars’ heights show the accuracy per predicted outcome. The other bars show the misclassification shares. As the chosen classes are time intervals, the algorithm makes a worse misclassification when predicting a class that is not close to the actual class. For example, in the first bar chart in figure 4.6 it can be seen that the algorithm performs very badly. Instead of predicting the correct class, "> 2 days early", it predicts "> 10 days late" which could have severe implications. If the algorithm had predicted "on time" instead, the damage would have been reduced. The numbers above the bars show the number of instances.
4.5. REFINING THE CHOSEN ALGORITHM

Table 4.9: The confusion matrix with initNaïveBayes set to true.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A: &gt;2 days early</td>
<td>0</td>
<td>99</td>
<td>14</td>
<td>0</td>
</tr>
<tr>
<td>B: On time</td>
<td>0</td>
<td>6578</td>
<td>306</td>
<td>0</td>
</tr>
<tr>
<td>C: 1-10 days late</td>
<td>0</td>
<td>1046</td>
<td>1761</td>
<td>0</td>
</tr>
<tr>
<td>D: &gt;10 days late</td>
<td>1</td>
<td>41</td>
<td>137</td>
<td>21</td>
</tr>
</tbody>
</table>

Figure 4.6: The classification accuracy with initNaïveBayes set to true.
Table 4.10: The confusion matrix with initNaïveBayes set to false.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A: &gt;2 days early</td>
<td>13</td>
<td>85</td>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>B: On time</td>
<td>9</td>
<td>6557</td>
<td>312</td>
<td>6</td>
</tr>
<tr>
<td>C: 1-10 days late</td>
<td>3</td>
<td>750</td>
<td>2027</td>
<td>27</td>
</tr>
<tr>
<td>D: &gt;10 days late</td>
<td>0</td>
<td>56</td>
<td>61</td>
<td>61</td>
</tr>
</tbody>
</table>

Figure 4.7: The classification accuracy with initNaïveBayes set to false.

When comparing figure 4.6 and 4.7, it is evident that the algorithm performs much better in general when initAsNaïveBayes is set to false. The algorithm especially improves in classifying the unusual classes, i.e., "more than 2 days early" and "more than 10 days late". Since the y-axes are showing percentage, it is easy to be deceived when looking at the charts. The numbers above the bars sometimes say more than the percentage. Many more instances are correctly classified when setting initAsNaïve to false. When the algorithm has not chosen the correct class for an instance it has most often chosen the nearest class, which minimizes the misclassification cost.
4.5. REFINING THE CHOSEN ALGORITHM

4.5.1 Transforming multiple classes to binary

According to Frank, Hall, and Witten (2017, p. 322-328) some algorithms perform better when a multiclass problem is decomposed into several binary class problems. This is due to the fact that learning is easier with fewer classes. There are two simple methods to transform a multiclass problem into a binary one. The first one is to produce several two-class datasets by letting one classification stand against the combination of the rest of the classifications. This method is often called "one-vs-rest". The second one is to use "pairwise classification" where a classifier is built for every pair of classes using only the instances from these two classes. The classifiers then vote on one of the classes and the class with the largest total amount of votes gets picked. There are algorithms for this in Weka that are ready to use. We tried them both against our previous results on 10000 instances from the dataset for Supplier 1. Both algorithms performed significantly worse than when not using this kind of algorithms. The "one-vs-rest" method got an accuracy mean of 85.51% and a standard deviation of 0.94. The "pairwise classification" method got an accuracy mean of 85.72% and a standard deviation of 0.95. Since both results were significantly worse, we decided not to use any of the methods.

4.5.2 Choosing search algorithm

Each of the search algorithms described in 4.1.1 were tested in Weka, on 10000 instances from Supplier 1. When testing K2, which uses a greedy search, and TAN, which generates the maximum weight spanning tree, the highest accuracies were obtained when applying the algorithm. The time complexity for K2 is $O(m \ast u^2 \ast n^2 \ast r)$, where $n$ is the number of nodes, $u$ is an upper bound on the number of parents a node may have and $m$ is the number of instances. Since $m$ is significantly larger than the other factors in the time complexity, the time complexity can be approximated to $O(m)$ (G. Cooper and Herskovits 2013). The time complexity for TAN is $O(m \ast n^2)$, where $n$ is the number of the attributes and $m$ the number of instances (Zhang and Ling 2001). As in the case with K2, the time complexity becomes linear in $m$.

TAN had an accuracy of 86.60% compared to K2 which had an accuracy of 86.55%. Since the time complexity is essentially the same for both K2 and TAN, and no search algorithm is significantly better than the other, we decided to continue with K2, since this is the search algorithm by default.

4.5.3 Discretization of continuous variables

According to Frank, Hall, and Witten (2017, p. 287) discretization of numeric attributes is essential when the algorithm can only handle nominal values, and even learning schemes that can handle numeric attributes often get better results or perform faster when the attributes have been discretized. As stated in section
4.1.1, the algorithm for Bayesian network in Weka prediscretizes the numeric attributes. As to find out whether a better result could be obtained with another discretization method, we tested different discretization filters available in Weka on different attributes. Weka differs between discretizing a numeric attribute and converting a numeric attribute to nominal. These two methods are represented by two different filters, henceforth called the "discretization filter" and the "numeric-to-nominal filter". For example, if one wants to discretize the value of OrderMonth it would be intuitive if the filter understood that there should be 12 nominal alternatives. In such situation the "numeric-to-nominal filter" is suitable. On the other hand, when discretizing for example DmdPrice which is more of a continuous variable with no obvious discrete intervals one should instead use the "discretization filter". We tried to discretize OrderMonth and OrderWeekday using the "numeric-to-nominal filter" and discretizing the other numeric attributes using the "discretization filter".

The best result was achieved when using only the "numeric-to-nominal filter" on the attributes OrderMonth and OrderWeekday, which gave an accuracy of 87.26% and a standard deviation of 1.05. This was significantly better than the previously best result.

### 4.5.4 Handle class-unbalanced data

As mentioned in section 3.3, class-unbalanced data could be a problem for Bayesian networks. Since the available data for Supplier 1 was very unevenly distributed, this was investigated. Weka has a function called under-sampling, which evens out the distribution of the classes in a dataset by removing instances of the most represented class. When using under-sampling and setting the relationship between the instances for the most and least represented classes to 5:1, the accuracy was 70.48%. The algorithm could handle the least represented classes more easily than before, as can be seen in table 4.11. The reduced number of instances when using under-sampling is due to the fact that the function has removed instances in order to obtain the specified relationship.

<table>
<thead>
<tr>
<th>Setting</th>
<th>#Instances</th>
<th>Accuracy &gt;2 days early</th>
<th>Accuracy &gt;10 days late</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without under-sampling</td>
<td>10000</td>
<td>11.5%</td>
<td>34.27%</td>
</tr>
<tr>
<td>With under-sampling</td>
<td>1443</td>
<td>35.5%</td>
<td>52.5%</td>
</tr>
</tbody>
</table>

If the customer would be more interested of knowing the outcome in extreme cases than having a very accurate algorithm, it could be a good idea to balance the classes.
4.5. REFINING THE CHOSEN ALGORITHM

4.5.5 Additional attributes

To further improve the performance of the algorithm we tried adding new attributes. This section describes the attributes that we tested. As using the "numeric-to-nominal filter" on attributes such as month and weekday and setting initAsNaïveBayes to false improved the accuracy, we will from now on keep these settings in every experiment.

Adding delivery month and delivery weekday

So far the month and weekday attributes that have been used have referred to the order date. Now we want to analyze whether the month and weekday of the demanded delivery date has an effect on the result. We call the two new attributes:

- \textit{DmdDeliveryWeek}: Represents the day of the week that the order is demanded to be delivered. According to some interviewees, the delivery pattern might differ between different days of the week, which made this interesting to analyze.

- \textit{DmdDeliveryMonth}: Represents the month in which the delivery is demanded to take place. A supplier could have a more difficult time to deliver in time if having lot of seasonal variations.

When running the attribute selector with all the previous attributes along with these two new attributes, on 10 000 instances from the dataset of Supplier 1, the algorithm only chose DeliveryWeekday in 10% of the cases and it never chose DeliveryMonth. When doing the same test on the dataset of Supplier 2 DeliveryWeekday was never chosen and DeliveryMonth was chosen in 10% of the cases.

The low values for DeliveryMonth and DeliveryWeekday indicate that the accuracy of this algorithm will probably not increase when adding these attributes to the supplied dataset. On the other hand, that is not reason enough to exclude the attributes entirely. This is just the case when looking at these specific customer-supplier relationships. Other customer-supplier relationships might get a much better accuracy using these attributes, which became evident when doing random samples on other customer-supplier relationships. Therefore the attributes DmdDeliveryMonth and DmdDeliveryWeekday will from now on be part of the group of attributes that the AttributeSelectedClassifier can choose from.

Adding site

Many customers have several sites to which their suppliers deliver products. A site can for example be one factory. It is likely that there are correlations between which site that is ordering the product and the supplier’s ability to deliver on time. This could for example be due to large differences in the physical distances between the
CHAPTER 4. APPLYING AI TO ORDER PLACEMENT

supplier and the different sites. We tried adding site, from now on called SiteKey, as an attribute and ran the attribute selector in Weka with 10-fold cross validation to see if the algorithm chose site to be part of the analysis. We ran the test using the dataset of Supplier 2 since the orders were made for two different sites in the historical data, in contrast to Supplier 1 who had only delivered to one site. The results are shown in table 4.12.

Table 4.12: The result from evaluating the attributes, including SiteKey, with BayesNet on all instances from the dataset of Supplier 2, using 10-fold cross-validation. The numbers show the share of the times in which the attribute was a part of the selected subset.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Share</th>
</tr>
</thead>
<tbody>
<tr>
<td>DmdPrice</td>
<td>60 %</td>
</tr>
<tr>
<td>DmdQty</td>
<td>0 %</td>
</tr>
<tr>
<td>ProdId</td>
<td>0 %</td>
</tr>
<tr>
<td>SiteKey</td>
<td>100 %</td>
</tr>
<tr>
<td>DmdDeliveryMonth</td>
<td>40 %</td>
</tr>
<tr>
<td>DmdDeliveryWeekday</td>
<td>0 %</td>
</tr>
<tr>
<td>NbrOfDays</td>
<td>100 %</td>
</tr>
<tr>
<td>OrderLinePrice</td>
<td>100 %</td>
</tr>
<tr>
<td>OrderMonth</td>
<td>100 %</td>
</tr>
<tr>
<td>OrderWeekday</td>
<td>100 %</td>
</tr>
</tbody>
</table>

As can be seen in the table SiteKey was chosen in 100% of the cases, which means it is probably an important attribute when the supplier delivers to several sites. We ran the AttributeSelectedClassifier on the dataset with SiteKey and compared it to running the algorithm on the dataset without SiteKey. The performance when using SiteKey, which had a mean of 70.10% and a standard deviation of 1.98, was significantly better than the result when not using SiteKey, which had a mean of 67.35%. Therefore the attribute SiteKey will from now on be part of the group of attributes that the AttributeSelectedClassifier can choose from.

Adding change boolean

Sometimes customers change the already placed orders. For example they might alter the ordered quantity or the desired delivery date. This could potentially affect the supplier’s ability to deliver on time. There are different potential implications of this. For example, if a customer wants a lower quantity than originally ordered, this could improve the supplier’s delivery ability, if having little on stock. On the other hand, it could also decrease the ability, since changes in the order lead to administrative tasks. Different changes can affect the supplier negatively or positively. Either way, it is likely to affect the supplier’s behavior. The best way to take changes into consideration would be to incorporate how and
4.5. REFINING THE CHOSEN ALGORITHM

when the order line was changed. Unfortunately, these changes were hard to follow in the available data.

There is a boolean called CustChange in the data base which is set to true if there has been a change in the order line and false otherwise. It would at least be interesting to add CustChange as an attribute in order to see how it affects the suppliers performance, if at all. As the number of order lines that had been changed in the database was very small, we could not investigate this any further. Instead, we chose to keep removing the order lines that had been changed from the training dataset, as mentioned in section 4.2, in order not affect to the results of the algorithm. Since the recommendation will be given when an order is initially placed, we only want the training dataset to contain instances representing the initial state.

Adding forecast

Customers often have some kind of plan telling how they will consume products or materials that they have a high consumption of. Generally, customers generate a forecast regarding which product and quantity they will need and when they will need it. This forecast is generated in advance so that they themselves and possibly also their suppliers know approximately what to expect in the near future. Some customers generate good forecasts that consort well with what they actually order later and some customers’ forecasts consort badly with the real demand. This can be measured with forecast accuracy, which shows the share of forecasts that were correct. It seems natural that a supplier would have better conditions to deliver on time if the forecasts are very accurate. Therefore it would be interesting to include one or several attributes that describe how a placed order differs from the forecast. Unfortunately it proved difficult to map a placed order to a specific forecast, partly because some of the desired data is not saved in the database and partly because some forecasts are only weekly or monthly estimates and not connected to one specific order.

It is possible to simulate the desired data and test the algorithm on the simulated data to see if the algorithm finds any correlations between forecasts and the supplier’s ability to deliver on time. In this case one would need to simulate one dataset where there is a pattern for the algorithm to find and one dataset where there is not a pattern for the algorithm to find. If the algorithm finds a pattern where there is one and does not find a pattern where there is none, it can be assumed that if one would like to use forecasts as an attribute in the future it can be done. Nevertheless, it seemed unreasonable to simulate as many attributes as would be needed if simulating the connection between a forecast and an order. If a very large part of the used data is simulated, it is hard to draw any valid conclusions from it.

As an alternative to adding several attributes that show how the forecast connected to the order looked, one could instead add just one attribute which shows only the current forecast accuracy of the customer. Maybe this could give the algorithm some indication that leads to a higher performance. It is also more realistic, since there is quite a big possibility that companies store records of their forecast accuracy,
in order to measure how well they perform. It is less likely that they store how a forecast has proceeded to an actual order. Since there was some trouble with acquiring this from the data base, a simulation was made in order to find out if the algorithm would notice any patterns in the data when adding forecast accuracy as an attribute. Rather than simulating all values, 10 000 order lines from Supplier 1 were used in this simulation, as to not ruin the patterns in the existing data. We have assumed that a higher forecast accuracy gives the supplier better conditions to deliver on time, since it has had more time to prepare for the order. We have simulated this by allocating random forecast accuracies to each order line depending on which class the instance eventually ended up in. The random numbers were normally distributed with a standard deviation of 10%, but had different means depending on the real class of the instance, as can be seen in table 4.13.

<table>
<thead>
<tr>
<th>Real class</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>More than 2 days early</td>
<td>50%</td>
</tr>
<tr>
<td>On time</td>
<td>75%</td>
</tr>
<tr>
<td>1-10 days late</td>
<td>50%</td>
</tr>
<tr>
<td>More than 10 days late</td>
<td>75%</td>
</tr>
</tbody>
</table>

The obtained accuracy when using the simulated values was 94.17%. This implies that the algorithm could find patterns in this attribute. We also did a simulation where the forecast accuracy was totally random, to be able to compare the two tests to each other. In this test, the algorithm did not choose forecast accuracy as an attribute. This indicates that the algorithm did not find any patterns in the attribute, which is favourable. The tests with simulated forecast accuracy illustrate that an attribute representing this could be added in the future without any major risks.

Adding year

The behavior of a supplier or a customer might differ between different points in time. A supplier that performed badly five years ago does not necessary perform badly today. Therefore it seems legitimate to have an attribute representing the year that the order was placed, henceforth called OrderYear. Then the algorithm might understand that it should put more weight on instances in the test data that are closer to the present. When testing if this attribute would have a large impact on the performance of the algorithm on the dataset of Supplier 1, the algorithm did not choose OrderYear as an attribute. This was probably due to a minor amount of instances with different years. Despite this, the attribute could be of importance in the future when there are more instances with different years. Therefore OrderYear will be part of the group of attributes that the AttributeSelectedClassifier can choose from.
4.5.6 Grouping suppliers

As the database contains many suppliers from which the customer only orders a few times per year, the algorithm might struggle with finding patterns for these. Therefore, it could be a good idea using the same algorithm for all suppliers, and having the supplier ID as an attribute. The algorithm might find suppliers behaving in similar ways, which could improve the accuracy. The distribution for the dataset when using all suppliers can be seen in figure 4.8. The accuracy when applying ZeroR on this dataset was 43.67%.

![Figure 4.8: The distribution of the real class values for all suppliers to the test customer.](image)

When trying to group all suppliers together and apply the algorithm with SupplierId as an attribute, an "Out of memory exception" was generated due to the heap size being too small. After trying several different configurations, we tried removing the attribute ProdId. This attribute is nominal and since all supplier offer a lot of different products, this nominal attribute could attain a staggering 38,500 different values. This turned out to be what had caused the "Out of memory exception" and the algorithm could be tested with all suppliers without the attribute ProdId. Random samples were taken to check how the accuracy for individual suppliers had changed, if at all, when grouping the suppliers together. These can be seen in table 4.14. The algorithm had an accuracy of 71.19% when grouping the suppliers.
Table 4.14: The accuracy when the algorithm is looking at one supplier individually versus grouping all suppliers. Supplier 1 and 2 are the same as previously, while the rest were randomly selected. The standard deviations are given within parenthesis.

<table>
<thead>
<tr>
<th>Supplier</th>
<th># Instances</th>
<th>Individually</th>
<th>Grouping</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10000</td>
<td>86.48 % (0.80)</td>
<td>82.45 % (2.34)</td>
</tr>
<tr>
<td>2</td>
<td>3958</td>
<td>70.12 % (1.98)</td>
<td>62.92 % (3.30)</td>
</tr>
<tr>
<td>3</td>
<td>3308</td>
<td>49.81% (2.50)</td>
<td>45.31% (2.89)</td>
</tr>
<tr>
<td>4</td>
<td>220</td>
<td>58.93% (6.22)</td>
<td>71.68% (11.16)</td>
</tr>
<tr>
<td>5</td>
<td>125</td>
<td>72.84% (5.00)</td>
<td>72.20% (14.29)</td>
</tr>
<tr>
<td>6</td>
<td>97</td>
<td>84.44% (9.02)</td>
<td>87.94% (9.51)</td>
</tr>
<tr>
<td>7</td>
<td>40</td>
<td>47.75% (8.02)</td>
<td>54.79% (25.62)</td>
</tr>
<tr>
<td>8</td>
<td>24</td>
<td>46.33% (32.70)</td>
<td>51% (33.60)</td>
</tr>
<tr>
<td>9</td>
<td>7</td>
<td>83.33% (37.58)</td>
<td>69.81% (40.34)</td>
</tr>
<tr>
<td>10</td>
<td>125</td>
<td>78.12% (5.32)</td>
<td>60.09% (10.41)</td>
</tr>
</tbody>
</table>

In six out of ten cases, the accuracy became worse when grouping all suppliers. The standard deviation increased for all samples. It seems that the individual suppliers have different patterns that the algorithm has a hard time finding when using the same model for all suppliers. Since the algorithm chose different attributes for different suppliers, it is interesting to look at which attributes that were chosen when using all suppliers. The algorithm only chose SiteKey, SupplierId, NbrOfDays and DmdDeliveryMonth. This makes it hard for the algorithm to find patterns on a more detailed level.

4.5.7 Additional features

In this section the implementation of the additional features that were mentioned in section 2.2 is discussed.

Predicting quantities

As delivery precision is measured as the share of orders arriving on time and in the correct quantity, an algorithm predicting the probable quantity that will be delivered seemed natural to implement. By combining the prediction of delivery time and quantity, it is possible to predict the delivery precision, which could be useful since it is a very common KPI. Four classes were distinguished: "more than 10 % less than the demanded quantity", "5-10 % less", "the demanded amount" and "more than 10 % above the demanded quantity". As with the previous algorithm, the classes of interest depend on the customer’s preferences and could be changed based on that. The attributes used were the same as before.
In 95% of the cases, when looking at all order lines of all suppliers, the delivered quantity was exactly as ordered. This could be due to the fact that the customer mostly orders very few items, which became evident when studying the data. The mean of the quantity for all order lines was 35.19, but most instances had quantities below 4.

The information gain for the customer when looking at the tested suppliers was presumed to be limited, since almost all order lines had the right quantity. Despite this, it could be a good idea to provide this function, since other customers could have more problems with this. By providing this function, the delivery precision KPI can be estimated.

Confirmation

When an order is placed, the supplier might confirm the order with a different quantity or delivery date than what the customer demanded. As an additional feature, the prediction of the delivery can be updated when the confirmation from the supplier arrives. This adds another layer of information to the customer. If the probability of the order arriving on the confirmed delivery date is low, this indicates that the supplier should be contacted as to find out where the problem lays. Why is the supplier committing to something they cannot deliver?

Since the supplier has confirmed the order, the attributes are somewhat changed as to reflect what the supplier has agreed upon. The attributes the algorithm can choose from are the following:

- **ConfirmedPrice**: The confirmed unit price of the product.
- **ConfirmedQty**: The confirmed quantity of the product.
- **ProdId**: The same as before.
- **ConfirmedMonth**: The month of the confirmation date.
- **ConfirmedWeekday**: The weekday of the confirmation date.
- **ConfirmedYear**: The year of the confirmation date.
- **OrderToConfirmDays**: The number of days between the order date and the confirmation date.
- **ConfirmToConfRcpDays**: The number of days between the confirmation date and the confirmed delivery date.
- **ConfirmedDeliveryWeekday**: The weekday of the confirmed delivery date.
- **ConfirmedDeliveryMonth**: The month of the confirmed delivery date.
CHAPTER 4. APPLYING AI TO ORDER PLACEMENT

- **ConfirmedOrderLinePrice**: The total confirmed price of the order line, hence it is given by:

  \[ \text{ConfirmedPrice} \times \text{ConfirmedQty} \]

One could think that it would be unnecessary to have both ConfirmToConfRcpDays and OrderToConfirmDays, but there could be a pattern for each. For example, if the supplier takes a long time confirming the order, it could be due to the fact that they have many other orders, or that they doubt they can deliver the order. A long time between confirmation and confirmed delivery date could indicate that there could be changes along the way, while a short time could indicate that the order will stay the same, since there is less time to change it.

When testing this algorithm it had an accuracy with a mean of 89.98% and a standard deviation of 0.67 when using 10,000 instances from the dataset of Supplier 1. As this particular supplier delivered as confirmed in 86.68% of the cases, the data was even more unevenly distributed than when looking at the demand data, which gave an even better prediction. This indicates that once a confirmation has been done, there are quite distinctive patterns in this supplier’s behavior. When testing on all instances from the dataset of Supplier 2, an accuracy with a mean of 63.20% and a standard deviation of 2.14 was achieved. For this supplier 65.71% of the orders were not delivered as promised, whereof 46.15% were before the promised date. When looking at the confirmation data the algorithm had an even harder time predicting the behavior of Supplier 2 than when looking at the demand data. This implies that Supplier 2 is very bad at keeping their promises. It is evident that in cases like this, an algorithm predicting the outcome based on the initial order could be of great value, as suggested in section 2.1.

Predicted delivery precision

It is possible that individual purchasers do not see the overall picture when the algorithm gives them a prediction. If the algorithm says that the order they are placing is going to be late, they will know this for just the particular order, and not for all the orders that are placed to that supplier. They could then have difficulties finding patterns in the supplier’s behavior. As analyzing the supplier often takes place on a more strategic level in the company and is mostly done using KPIs, it could be a good idea to implement a feature that compiles the information from the individual orders. This was also something that seemed interesting to the interviewed customers, since several of the interviewees pointed out that an indication of the supplier’s future performance would be of interest.

This overall picture could simply be shown by presenting the average probabilities for each classification out of all predictions that have been made for one supplier. The average probability of the orders arriving on time would then represent a kind of delivery precision.
5 | Final prototype and results

In chapter 4, several different choices of features were discussed. In order to clarify the final choices, a summarized prototype description is presented in section 5.1. An evaluation of the performance of the prototype is presented in section 5.2.

5.1 Specification

In the final prototype, it is possible to choose whether to make predictions regarding delivered quantity or delivery time. It is also possible to choose whether to make the predictions based on data from the initial order or based on the confirmation from the supplier. The choice of grouping suppliers or building separate models for individual suppliers can be made by the user itself. The classification algorithm used is BayesNet, with initAsNaïve set to false and maxNrOfParents set to three. The AttributeSelectedClassifier is used to choose an appropriate set of attributes. When predicting based on the initial order, the attributes shown in figure 5.1 are available.

![Figure 5.1: The available attributes when predicting based on the initial order.](image-url)
The attributes available when predicting based on the confirmation can be seen in figure 5.2.

![Figure 5.2: The available attributes when predicting based on the confirmation.](image)

Attributes representing days or months are discretized by using the "numeric-to-nominal filter". Other numeric attributes are discretized by the built-in discretization filter in the BayesNet algorithm.

It is possible to save the network structure to a file using Weka's own XML format called BIF. When classifying an instance, the network structure can be collected from the file. When initiating the network from the file, we discovered that the automatic handling of missing values and discretization did not work as intended, since the filters used for this were only initiated when building the model from scratch. Therefore, we had to change a few lines in Weka’s source code and make the initiation of these filters when reading the model from file as well.

It is assumed that no missing values will have to be handled by the model, since an order line naturally involves all the chosen attributes. The ability to handle missing values could be implemented in the future depending on the customer’s desires. It would be senseless to replace missing values with something else without knowing the specific customer. When classifying an order line that involves a new value of a nominal attribute the attribute is set to "new_value". Once the model is rebuilt or updated the new values are added to the possible alternatives for the nominal attributes.

### 5.1.1 User interface

The prototype has not yet been integrated with PipeChain’s system. To show how the results could be presented to the user when using the prototype algorithm, a
suggested user interface was implemented. To view predictions during order placement, the purchaser could choose to push a button called "prediction" in the customer’s business system, which calls upon the prototype algorithm. A window could then appear, showing the predicted probability for all the available classes for each order line. A suggestion on what this window could look like is shown in figure 5.3.

We chose to color code the classifications in order for the user to get a quick overview of the results. The color marked percentage on each line represents the most probable classification. A circle under the headline "Warning" indicates that it is less than 15 percentage units between the highest and the next to highest probability. The color of the circle shows the class with the second highest probability. This notifies the user that the prediction is uncertain. As to not confuse the user by showing predictions with very low probabilities, these could be prevented from being displayed. It would also be possible to only alert the user to view the predictions when the outcome is very certain and could potentially cause problems for the company. This could increase the likelihood that the user will react to situations that demand an action. If constantly being fed with predictions, the user might become indifferent to them.

A suggestion for how the delivery precision could be presented to the users has also been implemented and can be seen in figure 5.4. It is possible to choose to look at either the predictions based on the initial order or the confirmation. The KPI delivery precision is usually a combination of both time and quantity, but since some companies can have a bigger interest in knowing the accuracy in quantity and some in time, these are separated and the user can choose which one to look at. The person interested in the KPI also chooses a time frame, and the average probability of the classifications of all order lines that have a demanded delivery date during that time period will be shown. The average probability of the classification "on time" or "right amount" will indicate a kind of predicted delivery precision. This value is represented by the height of the green part of the bars and is also shown separately in percent in the table above the chart. The other parts of the bars show the predicted probabilities of the other classifications.
CHAPTER 5. FINAL PROTOTYPE AND RESULTS

Figure 5.4: Suggested GUI when presenting the KPIs. The patterns in the bars are only included in the report to increase the readability when printed in black-and-white.

The user could also receive a warning when the predicted delivery precision for a supplier is below a predefined limit. This would give the user an early indication without having to scrutinize the KPI chart every day.

5.1.2 Updating the model

If the system implemented during this project is integrated with PipeChain’s product in the future and thereby used by customers, it is important to update the model when new training data is available. It is generally preferable to avoid rebuilding the model from scratch, since this can be time consuming. In Weka it took about 22 minutes to build the model from scratch, using the AttributeSelectedClassifier in combination with BayesNet, on training data with 107,000 instances. Therefore it can be inconvenient to do this during the day when the program might be used. The BayesNet algorithm in Weka includes a function that enables updating the model by adding just one new instance. The AttributeSelectedClassifier does unfortunately not include such a function since it has to start its search through the space of possible attribute subsets from scratch. It is probably not necessary to update the group of selected attributes that often, though. One instance will presumably not affect the choice of attributes. Therefore the AttributeSelectedClassifier could be run to choose attributes over night, for example once every month and the BayesNet model could be updated with every new instance. If the AttributeSelectedClassifier chooses new attributes, the BayesNet model should be rebuilt from scratch during this nightly run.
5.2 Evaluation

The most important measure of the performance of the algorithm is the classification accuracy. We have mentioned the accuracy under many different conditions in this report. To summarize the accuracy of the final prototype the accuracy when running the algorithm separately on Supplier 1 and Supplier 2 along with the accuracy when using all suppliers are presented in table 5.1.

Table 5.1: The accuracy when building the model as specified by the final prototype.

<table>
<thead>
<tr>
<th>Supplier</th>
<th>ZeroR Mean</th>
<th>Prototype algorithm Mean</th>
<th>Std. dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supplier 1</td>
<td>56.94 %</td>
<td>86.48 %</td>
<td>0.80</td>
</tr>
<tr>
<td>Supplier 2</td>
<td>28.95 %</td>
<td>70.12 %</td>
<td>1.98</td>
</tr>
<tr>
<td>All suppliers</td>
<td>43.67 %</td>
<td>71.19 %</td>
<td>1.49</td>
</tr>
</tbody>
</table>

Considering the result of applying ZeroR to these datasets, the achieved accuracies are regarded as satisfying.

The final network structures when using the model on Supplier 1, Supplier 2 and all suppliers can be seen in figure 5.5.

Figure 5.5: The final network structures.
The confusion matrices for Supplier 1, 2 and all suppliers can be seen in table 5.2, 5.3 and 5.4. Each matrix is also shown in a bar chart, to better visualize the accuracy of the algorithm. This can be seen in figure 5.6, 5.7 and 5.8.

**Table 5.2: The confusion matrix for Supplier 1.**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>&gt;2 days early</td>
<td>21</td>
<td>82</td>
<td>10</td>
</tr>
<tr>
<td>B</td>
<td>On time</td>
<td>4</td>
<td>6558</td>
<td>303</td>
</tr>
<tr>
<td>C</td>
<td>1-10 days late</td>
<td>11</td>
<td>775</td>
<td>2023</td>
</tr>
<tr>
<td>D</td>
<td>&gt;10 days late</td>
<td>0</td>
<td>56</td>
<td>92</td>
</tr>
</tbody>
</table>

*Figure 5.6: The classification accuracy for Supplier 1.*
Table 5.3: The confusion matrix for Supplier 2.

<table>
<thead>
<tr>
<th></th>
<th>A: &gt;2 days early</th>
<th>B: On time</th>
<th>C: 1-10 days late</th>
<th>D: &gt;10 days late</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>967</td>
<td>94</td>
<td>55</td>
<td>30</td>
</tr>
<tr>
<td>B</td>
<td>196</td>
<td>777</td>
<td>133</td>
<td>22</td>
</tr>
<tr>
<td>C</td>
<td>141</td>
<td>212</td>
<td>718</td>
<td>73</td>
</tr>
<tr>
<td>D</td>
<td>69</td>
<td>23</td>
<td>137</td>
<td>310</td>
</tr>
</tbody>
</table>

Figure 5.7: The classification accuracy for Supplier 2.
Table 5.4: The confusion matrix for all suppliers.

<table>
<thead>
<tr>
<th></th>
<th>A: &gt;2 days early</th>
<th>B: On time</th>
<th>C: 1-10 days late</th>
<th>D: &gt;10 days late</th>
</tr>
</thead>
<tbody>
<tr>
<td>A: &gt;2 days early</td>
<td>4867</td>
<td>2367</td>
<td>1447</td>
<td>871</td>
</tr>
<tr>
<td>B: On time</td>
<td>1379</td>
<td>39067</td>
<td>4456</td>
<td>1919</td>
</tr>
<tr>
<td>C: 1-10 days late</td>
<td>1125</td>
<td>8103</td>
<td>15248</td>
<td>3683</td>
</tr>
<tr>
<td>D: &gt;10 days late</td>
<td>790</td>
<td>3642</td>
<td>3536</td>
<td>14722</td>
</tr>
</tbody>
</table>

When looking at the confusion matrices, it can be seen that the algorithm rarely classifies instances as the class furthermost from the correct class.

5.2.1 Self-awareness of the algorithm

The classification probability given by the algorithm can be interpreted as a kind of self-awareness of the algorithm. It is interesting to see how many times the algorithm were wrong given the predicted probability. An algorithm with a high self-awareness should be correct more often if the probability of the classification is high. The charts in figure 5.9 show histograms that illustrate this. The charts display histograms showing how many times the final algorithm was correct versus
5.2. EVALUATION

not correct given the predicted probability of the outcome. The algorithm was run 10 rounds with 10-fold cross-validation, using all of the instances from the different datasets. Each bar shows the number of instances with a predicted probability in the 10 percentage units large interval ending with the number showing under the bar. The charts to the left show only correctly classified instances and the charts to the right show only misclassified instances. The highest bars in the histograms of correctly classified instances should be close to 100 % and the highest bars in the histograms of misclassified instances should be of lower values.

![Classification histograms](image)

Figure 5.9: Classification histograms.

It is important to notice that the y-axes of the correct charts and the error charts are widely different. The charts with correctly classified instances has a y-axis reaching much larger numbers than the charts with misclassified instances. This means the vast majority of the instances are correctly classified, which consorts with the accuracy presented above. It can be seen from the charts of correctly classified instances that the number of instances increase with the predicted probability, which indicates a good self-awareness. The number of instances is very large when the predicted probability exceeds 90%, which means one should feel
quite safe to trust the prediction if the probability exceeds this. When looking at
the charts of misclassified instances it seems like the algorithm can sometimes be a
bit generous when suggesting a probability of the predicted outcome. On the other
hand, what would be most interesting to study is if the actual share of correctly
classified instances is highly correlated to the predicted probability given to the
classifications. To illustrate this we plotted the share of correctly classified
instances against the predicted probabilities for all instances from the dataset of
Supplier 1, Supplier 2 and the combination of all suppliers. The plots can be seen
in figure 5.10. The x-axes in the graphs start at 40% since there were very few
instances that had a predicted probability lower than 40%, which lead to
fluctuations in the graphs. In addition to this it is quite obvious that one should
not put much trust in the outcome when the probability of the classification being
true is that low, which makes these values very uninteresting. The straight lines in
the graphs are only present to show what would be a completely linear correlation.
5.2. EVALUATION

Figure 5.10: The charts show the percentage of correctly classified instances with respect to the predicted probability of the classification when running the final algorithm.

Since the curves are nearly linear in all the plots, one can assume that the actual share of correctly classified instances is highly correlated to the predicted probability given to the classifications. The fact that all the curves are nearly linear, even though the algorithm has quite different accuracy on the three datasets, indicates that the algorithm itself knows when it performs well and when it performs badly. The fact that the curve when using all suppliers is so close to the linear line is probably due to the large amount of data, which increases the probability of overshoot values to be extinguished by undershoot values.

5.2.2 The robustness of the prototype

As to test the robustness of the algorithm, noisy test data was inserted to see how the algorithm would react. The model was built on all instances from Supplier 1, and 5000 test instances were randomly generated. The probability that the test instances would contain any patterns were therefore slim. The quantity was set to a random number between 1 and 5000, the price between 1 and 20000 SEK, year as either 2014, 2015 or 2016 and NbrOfDays as a number between 1 and 100. All month attributes were set to a random number between 1 and 12 and all weekday attributes were set to a random number between 0 and 6. The OrderLinePrice was set by multiplying the simulated price and the simulated quantity. The nominal attributes SupplierId and SiteKey were randomly set to one of the available options. The "real classes" of the simulated instances were also randomly generated to be able to obtain the accuracy of the simulated test.

When testing this in Weka, the algorithm had an accuracy of 25.06%. This seems reasonable, since there are four possible classes to choose from. In 94% of the cases the algorithm predicted the most common class in the training data, "on time", which makes it very similar to ZeroR. The accuracy of 25% is simply due to the uniform distribution of the "real classes" in the simulated data.
5.2.3 Overfitting

To see if a model is overfitted to the training data one can classify the training data itself to see what accuracy is achieved. This was tested on the whole dataset of Supplier 1 and Supplier 2 and the result presented in table 5.5 was achieved.

Table 5.5: The result when classifying the training data from Supplier 1 and 2.

<table>
<thead>
<tr>
<th>Supplier</th>
<th>Classifying training data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supplier 1</td>
<td>86.22%</td>
</tr>
<tr>
<td>Supplier 2</td>
<td>75.16%</td>
</tr>
<tr>
<td>All suppliers</td>
<td>74.61%</td>
</tr>
</tbody>
</table>

The accuracy for Supplier 1 is about the same as when doing 10-fold cross-validation. The accuracies for Supplier 2 and when grouping all suppliers are somewhat higher, but not unreasonably high. This indicates that the model is not overfitted. If the model had been very overfitted, the accuracy when classifying the training data would have been near 100%. As stated previously, the standard deviation can be viewed as a measure of overfitting. Since the standard deviations are quite low, this does not seem to be the case. Due to these two indicators and the fact that Bayesian networks are not that prone to overfitting, it was concluded that overfitting is not a problem when using the prototype.
6 Discussion

In this chapter we present a critical review of our own work. We discuss difficulties that we have encountered, benefits and limitations of the prototype and risks of fully trusting the outcome of the algorithm.

6.1 Difficulties

One of the major difficulties during this project has been collecting data that could be used to build and analyze the AI model. PipeChain’s access to customer data is limited and therefore we have not had the opportunity to be selective when choosing which customers to look at. If we would have had the opportunity to pick customers that had a certain kind of data, the analysis might have been richer. In this case we were simply compelled to look at the customers that had any available data at all. This made some of the subjects that we wanted to investigate hard to study. For example it would have been interesting to consider order change in the analysis. Changing an order line might have great impact on the supplier’s ability to deliver as demanded. A change in the order line could have either positive or negative effects on the supplier, depending on what has been changed. Therefore it is likely that the model needs information on what change has been made in order to make better predictions. Out of the available data almost no order lines had been changed, which made this kind of analysis very hard to perform. In addition to this, some order line history was missing. The order change date was also saved on order level and not per order line, making it hard to track when a certain line had been changed. Therefore we have not had the opportunity to analyze how order changes influence the delivery outcome.

The same reasoning can be applied to the analysis of forecasts. This would have been very interesting to look at and could probably have a great impact on the accuracy of the algorithm, but it was not possible to analyze with the available data. It was for example not possible to follow an order from forecast to placement, since forecasts are often made on a weekly or monthly basis.

As was seen in section 4.3.2, the distribution can vary a lot between different suppliers. An uneven distribution makes it easier for the algorithm to predict the most common class, but also makes it harder to predict the most rare class. The question is, which classes are considered most important to be able to predict?
CHAPTER 6. DISCUSSION

This is very individual to each customer and is therefore hard for us to discuss. Nevertheless, it is important to pinpoint that customers that mainly want to predict the most rare classes might need a slightly modified algorithm that balances the distribution between the classes. There are functions in Weka for this, that either remove instances from the over-represented classes or add simulated data to the under-represented classes. One of these was tested in section 4.5.4, but it is not used in the final prototype since different customers might have different interests.

6.2 Seasonal variations

As pointed out by the consultants, many customers face seasonal variations. Unfortunately, most of the order lines in the data base were made during a single year, making it difficult to analyze these variations properly. Despite this there are monthly attributes incorporated in the final prototype which try to look at seasonal patterns. These have proven to be important attributes which have been chosen for many of the suppliers. There could potentially arise a problem when using these attributes in the model, though. Let us assume that a supplier starts their improvement work in January and that the improvements increase their delivery precision linearly during the year. Since the supplier is constantly improving, each prediction will slightly underestimate their performance. The biggest problem will arise when a new year is commenced. If the algorithm takes month into consideration, it will believe that the supplier generally performs badly in January. But in reality the supplier will probably perform at least as well as in December the previous year. It is hard to say how this problem is affected by the lack of historical data. It is possible that the algorithm would be better at handling these changes when having several years of data available since year is also an attribute, which could make it understand that the supplier’s performance is better this year than the year before. But it is very difficult to say how the algorithm would react. One way of reducing the impact could be to either not use monthly attributes the first couple of years, or to accept that the algorithm will handle improvement work badly. Another way could be to make the suppliers, at least the ones delivering more strategic products, tell the customer when improvement work will take place. This could make it possible to incorporate it in the model beforehand, by for example reducing the weight of the monthly attributes. It could also be possible to look at the previous years’ KPI of the delivery precision and alter the model accordingly.

6.3 Limitations of the prototype

The attribute ProdId was not included in the final prototype. It could be a drawback for some customer-supplier relationships, but when doing random tests, ProdId was not chosen for many of the tested suppliers. This could have a quite intuitive
6.4. THE BENEFITS OF THE PROTOTYPE

explanation. As stated before, one wants to keep the algorithm as simple as possible and the attributes chosen should be valuable to the algorithm. In a customer-supplier relationship where the customer buys a wide range of different products, the nominal attribute ProdId will of course also have an equally wide range of alternatives. This means that the attribute would only increase the complexity of the network structure and the computational work needed, since the probability tables would be large. There would also be very few instances for each value of ProdId in comparison to the amount of available data, especially when grouping the suppliers. It is therefore quite reasonable to imagine that even if the model would have been able to handle the attribute ProdId when grouping all suppliers, it would probably have been discarded by the AttributeSelectedClassifier. The interviews with the customers also indicated that it could be a good idea to look at supplier level and not product level, since they often had problems with certain suppliers and not specific products. This supports our belief that ProdId should not be a part of the final product.

The algorithm performed slightly worse for some of the suppliers when grouping them in contrast to when treating them individually, which can be seen in 4.5.6. It could be a question of needing more data in order for the algorithm to be able to separate between patterns of different suppliers. The lower performance could also be due to the fact that the algorithm chooses other attributes than what is optimal for the individual supplier. The suppliers with a small amount of data could on the other hand gain from combining the suppliers. But one must be prepared for the possibility of a lower accuracy and a higher standard deviation, at least before larger amounts of data are available.

The final product cannot handle missing values. If an order that lacks one of the attributes is placed, a prediction cannot be made. This has been made intentionally, since we believe that an order not containing one of these attributes should not be possible to make. Since many of the attributes are derived from others, the only attributes that the purchasers actually need to fill out are DmdQty, DmdPrice, SiteKey, SupplierId and demanded delivery date, since order date is generated automatically. If one of these are missing values, an exception should be generated since the purchaser has made an incomplete order. If a customer does not use one of these attributes, it is better to make a customized version for this customer.

6.4 The benefits of the prototype

There are many positive effects that can be obtained from the final prototype which can help both suppliers and purchasing companies. The purchasers get an early indication of when the products will arrive, which can be sent to the production department. It could also be possible for the purchasing company to give promises to their customers that they are certain they can keep, for example by prolonging the promised delivery time for a customer product when orders are predicted to be
delayed. They could also get an insight into how their behavior affects the supplier, by analyzing the attributes chosen by the AttributeSelectedClassifier. If for example OrderWeekday is an important attribute, the company could analyze if orders placed on a certain day always cause trouble. The supplier on the other hand could get purchasers that are more tuned to their business and understand when an order obstructs the work of the supplier. Hopefully in the long run, the kind of behavior that puts the supplier in distress might be avoided. If the algorithm has a hard time giving predictions for a certain supplier, even this could be valuable information for the customer. If the supplier show no reasonable pattern in their behavior, maybe they are not a suitable partner to work with. A discussion could be held with the supplier in question, in order to find out why the odd behavior is taking place.

While studying the data, many instances were discovered where the delivery date was set to a date before the day when the order was placed. This is probably made in order to signal to the supplier that that order is very urgent, as to get prioritized. This can affect the KPI for delivery precision, if measured on the basis of demanded delivery date, making the strategic purchasers believe that there is a serious problem with the supplier. At the same time, the supplier might see the purchaser’s behavior as very odd and get frustrated by receiving these emergency orders as well as a bad KPI. When implementing a model like the suggested one, it would be very counterproductive to place an order of this sort, since the probability of it arriving on time would be zero, which might reduce this behavior.

The model could be used in order to pinpoint future problematic suppliers by studying the KPI graphs. If building individual models for each supplier, an indication to the source of the problem could also be made by looking at which attributes that are chosen. An analysis of the supplier’s future behavior can of course be made by an analyst, where a trend curve for the past year’s performance can be made and conclusions can be drawn from this. It is difficult for an analyst to keep all relevant things in mind though and they might miss some patterns in the data. It is also very time consuming to manually analyze the supplier’s performance, at least when using large amounts of data. In addition to this, the manual analysis cannot be updated as often as the AI model could be. This makes it more dynamic than manual analysis. People with much experience within purchasing still have valuable information that can be combined with the model. These people know how to act based on the information presented to them.

6.5 Suitable conditions of use

There are some conditions under which the algorithm operates optimally. As mentioned above, the algorithm cannot handle missing values and it is therefore important that the customers using it fill in all the necessary attributes when placing orders. The algorithm has a better accuracy when the data is unevenly distributed since this makes the most common classes much easier to classify. As mentioned above this leads to the algorithm having a harder time predicting the
most rare class, though. The most critical prerequisite needed to make good predictions is the presence of statistical patterns in the suppliers’ behaviors. It is likely that the algorithm will improve when more data is available. Therefore it is also important that the customers finish their orders, so that they can be used as training data.

If the system is ever integrated with PipeChain’s product, some customers might benefit more from it than others. The system is suitable for purchasing companies that want to analyze the behaviors of their suppliers. In some industries, like the automotive industry, the technical handling of orders is highly developed and in other industries, like the construction industry, many orders are still placed via telephone calls. It is important that the user has implemented a system for order placement that reaches an adequate technical level. A base requirement is of course that all orders are placed via PipeChain’s system and not for example through telephone calls. The ability to predict the outcome of an order placement might also be of different interest in different industries. In some industries, for example producing industries where there is no stock, it is very important to know when an item will arrive, since it will be sent into production right away. In other industries where the inventory costs are low this might be of less interest.

6.6 Risks

Unfortunately, it is impossible to achieve an accuracy of 100% when implementing the desired algorithm. Therefore there is a risk in trusting the predictions completely. Luckily, the algorithm often predicts the nearest class to the correct one when being wrong, which should reduce the consequences of trusting it. As mentioned in section 5.2.1, the algorithm is quite aware of when the outcome is more or less likely. Since the user is given the probabilities for each interval they can take these into consideration before making a decision based on the prediction. If a full decision support system would have been implemented the risks would have been much higher, since the algorithm would suggest a single action to take. The predictions made by the algorithm are hints of what could happen and could be used as a base when making decisions. Nevertheless, it is still important that the user has domain knowledge and can understand when an action needs to be taken.
7 | Conclusions

This report shows that techniques within artificial intelligence certainly can be applied to supply chain management. As our experimentation has shown, Bayesian networks can achieve a very good result, given data showing somewhat clear patterns. Also decision trees had a very good accuracy when it comes to classification. If one would like to predict something within the supply chain area without needing the probabilities for each class, this could be a great alternative to Bayesian networks. The combination of interviews, data analysis and experimental testing has made it possible to answer all research questions.

"What problems are present in today's order process and which of them are possible to solve with the aid of artificial intelligence?"

As found by the interviews, typical problems for purchasing companies using traditional order placement are suppliers not confirming the order, not delivering as confirmed or withholding information about problems. This leaves very little time for the purchasing company to adjust the production plan according to what components will be available. For suppliers using traditional order placement, irrational orders from the customers are often a problem. For example some customers set a delivery date that is in the past. A big problem when using VMI towards customers is handling inaccurate forecasts made by customers, which makes it hard for the supplier to plan their deliveries. It is not unreasonable to imagine that this might be a problem for suppliers not offering VMI as well. Bad forecasts have not been investigated during this project due to a limited time frame. It is evident that the problems emerging for both suppliers and customers are mostly due to a lack of information exchange between the two parties. The interviewees believed predictions, as the ones made by our prototype, could help their companies’ order placement process and to some extent decrease the problem of limited communication. Some operative purchasers have a limited understanding of the company’s strategic work. This has been defined as one of the problems with today’s purchasing. The predictions made by our prototype could help in guiding these people to make better decisions.

The prototype that was implemented during this project is only one example of how AI could be applied to SCM. During the project many other ideas, that we unfortunately have not been able to realize in the short time frame, have occurred. These are presented below in the Future work section.
"What benefits can the supplier and the purchasing company obtain by using a prototype like the one produced during this project?"

By using the model, the purchasing company can get a hint on how they should alter their production plan if necessary. This might lead to less down-time, which in the end saves money for the company. It would also be possible to give the end customers an early indication of the delivery time being longer, which might mean that customers will not be lost when deliveries are delayed. It could also pinpoint potential problematic suppliers. The attributes chosen by the algorithm indicate which factors that affect the supplier the most. This means that the purchasing company could have a possibility to learn more about their suppliers, and change their ordering behavior accordingly. The purchasing company might therefore be able to find problems in its own behavior that affects the supplier negatively. The supplier and purchasing company could therefore get an increased understanding of each others’ work and improve their relationship. In conclusion, the system could facilitate the customer-supplier relationship without needing to share more information.

"Which data and what amount of data is needed to make reliable predictions of the flow of materials in the supply chain?"

As this report has shown, several attributes are needed, and they differ between suppliers. This indicates that individual suppliers face different challenges. All final attributes, except for OrderYear, have been chosen by the algorithm in at least one of the customer-supplier relationships studied. It is natural that OrderYear has not been chosen since the majority of the available data was from a single year. We strongly believe that OrderYear will be an important attribute when data from several years is available. As discussed earlier the algorithm is quite self-aware of its own reliability, which means the predictions can be trusted when the given probabilities are high. Regarding the amount of data needed, it is obvious that more data leads to better predictions, but when reaching a certain amount the impact of more data is limited. The most important thing needed to be able to make reliable predictions are the patterns of the supplier’s behavior. The stronger the pattern, the less data is needed. But since very small datasets are prone to overfitting, these should be avoided. In this kind of system it is impossible to reach an accuracy of 100%, mainly due to the fact that one can not cover all factors that affect the outcome. There are many external events that might have an impact on the supplier. For example there could be labor strikes, global shortages of raw material or natural disasters. In addition to this there is always a human error that an algorithm cannot predict.

In this thesis, a subgroup of purchasers and consultants has been analyzed. We leave it to others to investigate what other stakeholders might desire. But this case study at PipeChain Group AB can be a hint on how to find the problems in purchasing and how to reduce them using artificial intelligence.
CHAPTER 7. CONCLUSIONS

Future work

Since it was not possible for us to analyze the effect of order changes and forecasts, this could be further investigated. It is possible that the algorithm could achieve a better result if these attributes could be used. The algorithm might also be improved if having access to supplier data. If the supplier is reluctant to share the data, it might be possible to interview suppliers in order to find out what factors that affect their delivery precision the most. In this project, we have not given any recommendations whether to group suppliers or analyze them individually. When grouping suppliers without looking at their individual patterns some information might be lost. It would therefore be interesting to group the suppliers according to commodities, usage of the same subcontractor or similar behavioral patterns and see how this affects the accuracy. This would demand a lot of manual work, but might have positive outcomes.

As mentioned above, there are many external factors that could potentially influence the supplier’s behavior. Therefore, it would be interesting to combine historical data with real-time data from the surrounding world. This could be conducted by for example analyzing online news flows.

The interviews with the consultants revealed that some purchasers lack an understanding of the tactical and strategic purchasing, which can lead to faulty decisions. Therefore, a decision support system might be of good use. The system could help the purchasers alter their orders by for example suggesting a quantity that increases the probability of the orders arriving on time.

As mentioned above suppliers using VMI towards their customers experience difficulties due to bad customer forecasts. The interviewees requested a system that can predict the customers’ material usage based on forecasts and historical data. Even though this has not been investigated during this project, we believe that a similar approach could be used to implement such a system.

Almost all problems found can be traced back to a lack of information exchange between the supplier and the purchasing company. This inhibits the optimization of the supply chain. It is understandable that companies do not want to share sensitive data with other actors on the market, since the data could be used in ill-meaning ways. Therefore the ultimate solution could be an artificial intermediary. This system could operate as an unbiased black box receiving sensitive data from both parties and giving recommendations on how the parties should cooperate. This way, an information exchange can take place without displaying the sensitive data to the other party.
8 | References

Articles


Books


Websites


Appendix
Appendix Contents

A Interview guide - customers 79

B Interview guide - consultants 81

C Interviews 83
   C.1 Company 1 ........................................ 83
   C.2 Company 2 ........................................ 84
   C.3 Company 3 ........................................ 85
A  |  Interview guide - customers

• What is your role as an employee at . . . [company name]?

• What industry do you work in?
  – Directly to end customers? Finished products?

• What does the ordering process look like?
  – Traditional purchasing, VMI . . .
  – Do you use different ordering processes for different items?
  – Do you order many different items in the same order?
    * Is the entire order delivered at once? What happens if one item is delayed?
    * If one item is delayed, is the entire order seen as delayed? How is your KPI for this measured? On orders or order lines?

• What does your production look like?

• What services do you use in PipeChain?

• Which suppliers are linked to your work?
  – Are all suppliers linked to PipeChain services?
  – Do you use several suppliers for the same component?
  – Do you have any problems with a particular supplier?
  – What kind of relationship do you have with your suppliers?

• When do you consider an order to be on time? The precise date or OK with early arrivals (and how early in that case?)

• Are there any seasonal variations? When?
  – Are there variations between weekdays?
  – Are there big variations in customer demand?

• Do you have any regular campaigns?
  – When was the last time you had a campaign?
• What are the most common problems you encounter in the ordering process today?

• Are there any specific situations where problems often occur?
  – For example, when ordering a certain item or within a certain time frame from delivery, the orders are often late

• We will use historical data in order to predict future events. What services/functions would you like to have, bearing this in mind?

• Would you value these functions...
  – Predicting whether an order is probable to arrive on time or not (maybe compare it to the use of having a percentage?)
  – Predicting that the order will be 1-3, 4-6 days late etc.
  – Action proposals
    * E.g., if you order 3 instead of 4, your delivery date will probably be...

• What variables do you think are important to include in the analysis?

• Considering the entire logistics industry, not just the ordering process: what do you believe would be the most attractive service using AI?
Interview guide - consultants

• What is your role as an employee at PipeChain?
• Which customers do you work with?
• What industries do the customers work in?
  – Directly to end customers? Finished products?
• What does the ordering process look like?
  – Traditional purchasing, VMI...
• What does the production process look like?
• What services do the customers use in PipeChain?
• Which suppliers do they work with?
  – Are all of them linked to PipeChain?
• Are there any seasonal variations? When?
  – Are there variations between weekdays?
  – Are there big variations in customer demand?
• For how long is the data stored? How old data is available?
• What are the most common problems you encounter in the ordering process today?
• Are there any specific situations where problems often occur?
  – For example, when ordering a certain item or within a certain time frame from delivery, the orders are often late
• We will use historical data in order to predict future events. What services/functions would you like to have, bearing this in mind?
• Do you think the customers would value these functions...
– Predicting whether an order is probable to arrive in time or not (maybe compare it to the use of having a percentage?)
– Predicting that the order will be 1-3, 4-6 days late etc.
– Action proposals
  * E.g., if you order 3 instead of 4, your delivery date will probably be...

• What variables do you think are important to include in the analysis?
• Considering the entire logistics industry, not just the ordering process: what do you believe would be the most attractive service using AI?
C  |  Interviews

This appendix describes information gained during the interviews with PipeChain’s customers more thoroughly. As the customers are anonymous they are simply called Company 1, 2 and 3.

C.1 Company 1

The first interviewee is a commodity manager and application manager at company 1, C1, and has worked at the company for six years. He is part of the strategic purchasing division, but has a good knowledge in the operative purchasing as well. His job is among other things to find and make contracts with suppliers and make sure they are performing well. C1 are producing finished products for businesses. The products are often highly customized, but have to some extent standardized components. They are active in a low volume industry and have many components that they only buy a few times a year. They only use traditional order placement, i.e., they do not have VMI with any of their suppliers. They have no significant seasonal patterns, and they try to use as few suppliers as possible.

C1 measure their KPIs per order line and consider an order line being on time if it arrives the same day as the demanded delivery date or a maximum of two days before. It is fine if the delivery contains a larger quantity than requested, but not less. Their biggest problem is suppliers who cannot deliver on time. When there is a high pressure on the suppliers in the marketplace, for example during a trade boom, the suppliers often struggle with the deliveries due to either problems with finding the material or having labor shortage. Most suppliers are sending order confirmations, but C1 have some problems with suppliers changing the delivery date without telling them. The interviewee does not see any seasonal patterns neither in the demand from the customers nor in how the purchasers are placing orders.

He believes it would be very interesting to have predictions within the ordering process. Currently, they have many problems with certain suppliers. The possibility to get an early indication or warning telling when a supplier has a tendency to deliver late, even just a week in advance, could help the company a lot. If the prediction could indicate how many orders that could be affected and how late they would be, that would be even better. A problem today is that the suppliers often withhold information about their problems as long as possible, in order to not lose face. When
the information finally arrives it is evident that the supplier has promised more than 
they can handle with their order confirmations, and that deliveries are very delayed. 
As to our suggested features, he was very positive to them if the predictions are 
shown to be somewhat accurate. He argues that depending on which division that 
is ordering, different intervals for the delivery date are interesting, since the divisions 
have different levels of safety stock.

C.2 Company 2

The second interviewee has more than 15 years of experience within supply chain 
management and currently works as Supply and Demand manager at company 2, 
C2. C2 is a multinational company which does not carry any inventory. The 
production is outsourced, which means that they mostly purchase finished 
products and configure these. The products are not customized, but they have a 
very big assortment. They use VMI for some of their suppliers and traditional 
order placements for the rest and they work with 150-200 suppliers in total. 
PipeChain is not used for all orders, but approximately 90% of their order value 
goes through PipeChain. They have chosen not to use PipeChain for the suppliers 
they only buy from a couple of times per year, since the information gain from this 
would be too small. C2 perceive an order as being on time if it arrives in the 
interval between 2-5 days early and 2 days late and the quantity should be as 
ordered. They have noticed that the suppliers are struggling to deliver according 
to the order when C2 has a significant demand above forecast. What outcome this 
will lead to is then much determined by what relation they have with the supplier 
and if they are prioritized as a customer. There are quite clear seasonal patterns 
which are taken into account when doing the prognosis.

C2 has a customer lead time, including transportation, of 6-10 days. Since the lead 
times from the component suppliers vary between 1-22 weeks, and some 
distributors at some regions are minimizing their inventories, a late delivery can 
cause a lot of problems. Unfortunately, it happens that the suppliers deliver late. 
It also happens that the suppliers confirm the orders, but fail on their promises. 
The interviewee would like to see an algorithm that could predict whether they 
should place orders earlier, for example when they are placing very many orders. 
Something that predicts a different order quantity for a better result or that 
adapts the lot size would also be very interesting. She seems interested in our 
suggested features, which are mostly related to traditional order placement, but 
argues that it would be more interesting to improve the VMI flow, since they are 
trying to reduce the traditional order placement.
C.3 Company 3

During this interview two interviewees, employed at company 3, C3, were present. One of them is the logistic manager. The other one is working for the planning department, where she has had several different roles. C3 is positioned at the beginning of the supply chain, which means there are many intermediaries between them and the end customer.

C3 uses PipeChain to run VMI towards some of their customers, which makes their response reflect the supplier’s view of using PipeChain for VMI. At the moment C3 has nine customers with whom they use VMI. They see it as a possibility to improve their relationship to these customers. The interviewees point out that the order suggestions given in PipeChain’s VMI system only function as guidelines for what orders to place. They explain that there is a lot of manual work adjusting the suggestions to real orders that can actually fulfill the customer needs. This is mainly due to the low forecast accuracy of the customers. C3’s customers have forecast accuracies between 20% and 60%, which makes the forecasts hard to trust. This obstructs PipeChain’s work of generating good order suggestions. The interviewees both mention that the dream scenario would be to get suggestions from PipeChain that are so good that they can make them real orders directly, without doing any manual work adjusting the suggestion. To get such good suggestions using the techniques included in PipeChain today, the forecast accuracy of the customers would have to be significantly improved. Unfortunately the customers have very few incentives to improve their forecasts. C3 often put a lot of effort into adjusting the given suggestions, but this is only a cost for C3, not for their customers. If C3 make it work with the bad forecast accuracy, why would the customers be interested in improving their forecasts? The customers know that they will get their products on time anyway. Because of this, the interviewees see a great possibility in using artificial intelligence to analyze how well a suggested order in PipeChain matches the real need of the customer and what uncertainties might affect the reliability of the suggestion. This could be done by also including the customer’s forecast patterns when running the algorithm.

C3 experience great variations in their customers’ needs. The needed quantity can vary a lot between different weeks or months. In addition to this C3 itself also have a harder time shipping products on certain days of the week. This is mainly due to the fact they want to match their use of trucks with other industries in the local area and that these industries get their deliveries on certain weekdays.
Förbättring av försörjningskedjan med hjälp av artificiell intelligens

Artificiell Intelligens, AI, har funnits det senaste halvseklet och har förbättrat många industrier. Trots goda resultat har inte AI använts fullst ut inom Supply Chain Management, SCM. Hur kan egentligen AI tillämpas på SCM och vilka är fördelarna?


Datamängden som behövs är intressant nog inte det mest avgörande för att göra säkra prediktioner, även om det såklart behövs en hel del data. Desto viktigare är att leverantören uppvisar ett visst mönster i sina handlingar. Beter sig leverantören helt slumpmässigt kommer inte modellen kunna förutspå hur den kommer att leverera. Detta är intressant nog även något man kan dra nytta av som kund. Uppvisar leverantören inget begripligt beteende kanske man inte borde arbeta med den?

Modellen kan alltså lösa många problem i dagens försörjningskedjor och skulle kunna förändra användningen av AI inom SCM. Om PipeChain Group AB integrerar modellen i sitt system kan denna förändring komma snart.