August 11, 2020

Master thesis

Predicting failure modes of traffic control systems by classifying unstructured data of maintenance requests

– Final version –

R. Haandrikman BEng. (Robin)

MSc Industrial Engineering and Management
Faculty of Behavioural, Management and Social Sciences (BMS)

UNIVERSITY OF TWENTE.

dynniq
energising mobility

COLOPHON

Document type	Master thesis
Title	Predicting failure modes of traffic control systems by classifying unstructured data of maintenance requests
Company	Dynniq Nederland B.V.
Author	R. Haandrikman BEng. (Robin)
University	University of Twente
Study	MSc Industrial Engineering and Management
Graduation committee	Dr. M.C. van der Heijden (University of Twente) Dr. Ir. L.L.M. van der Wegen (University of Twente) Ing. C. P. Heineman MSc. (Dynniq Nederland B.V.)
Keywords	Data Analysis, Data Mining, Text Mining, Classification Model, Decision Tree, Support Vector Machine, Artificial Neural Network, Preventive, Corrective, Maintenance, Failure Mode, Traffic Control System

MANAGEMENT SUMMARY

This report describes the research of ways to improve the maintenance of traffic control systems at Dynniq Nederland B.V. Dynniq is a company that helps to manage mobility flows in society through advanced technological solutions. The company sees the need to decrease their maintenance costs due to an expected decrease in revenue on maintenance contracts. The aim of this research is to explore what potential value is available in the historical data on maintenance requests and solutions, which will ultimately lead to a cost reduction. By finding this potential, we can answer the main question of this research:

"How can historical data on maintenance requests and solutions improve the effectiveness of any kind of maintenance, making it more future-proof?"

The research was conducted according to the Cross-Industry Standard for Data Mining (CRISP-DM) methodology, which includes the entire process from communication of the business problem through data collection and management, data preprocessing, model building, model evaluation, and model deployment. The main deliverable of the CRISP-DM methodology is an implementation plan for a designed model that answers the main research question of the research.

To find out what part of the company's performance has the best improvement potential, a performance analysis was executed using the available data set on maintenance requests and solutions. The outcome of this analysis was an improvement potential on inspecting unstructured text fields like *problem description, cause,* and *solution* to increase the knowledge on why systems fail, why some of those failures cannot be solved at first attempt, and why preventive maintenance does not reduce the number of failures per system. This information can be extracted from the unstructured text fields in the available data set by the use of text mining. Text mining transforms text into a mathematical form that can be used for calculations. In this research, we used text mining as an input for the built of a classification model. This model is based on the way systems can fail, also known as failure modes. The model uses the transformed text data to predict which class (i.e. failure mode) belongs to the problem description of a Service Order that should be solved. Service Order is the generic name for a work order of a malfunction in a system and the associated maintenance activities.

A literature review has been conducted to find appropriate methods for building a text classification model. Key findings from the literature review are a theory on failure modes and appropriate algorithms for building a text classification model. The chosen failure modes *electrical failure, software failure, mechanical failure, external damage, human error,* and *no problem* were based on the theory of Tinga (2013). The classification algorithms found in literature are an Artificial Neural Network (ANN), a Support Vector Machine (SVM), and a Decision Tree (DT). These algorithms are chosen for their ease of implementation, performance potential, and availability in free software packages.

Based on the findings from literature, a classification model was designed. The model serves as proof of concept to determine whether a text classification model can improve the knowledge on failure behaviour of the traffic control systems. The model was built and validated with a data set of 500 Service Orders. These Service Orders were evenly divided over ten different system types. The system types are systems placed at intersections and along highways that are produced and maintained by Dynniq. All Service Orders in the data set were classified in advance to be able to compare the prediction of the model with the predetermined class label. To determine how well the model performs, the data set was split into a training and validation set. The training set was used to build the classification model

and the validation set was run through the model to predict which class labels correspond to the Service Orders. The performance of the model was measured with an accuracy score, which can be defined as the percentage of Service Orders from the validation set that were correctly classified. The model provides a result that is presented as partial match, for example 60% chance of an electrical failure, 40% chance of a software failure. The higher the distinction between these class labels, the higher the chance that the best match is the actual failure mode. To optimize the model, the validation was executed for three different ways of partitioning the same data set of 500 Service Orders. In the first test, the data was split per system type in ten evenly divided groups. In the second test, the same data set was split in two system groups (i.e. highway and intersection systems). The third test had only one partition, consisting of all system types together.

The evaluation showed that the best performing partition of the data set is on system type, which means that each system type has its own model. For this partition, the average accuracy of the ANN (61%) and SVM (61%) models are higher than the DT model (55%). The accuracy of the ANN and SVM models are similar with a maximum difference of 6% per system type. As the ANN model is only the simplest form of a neural network, the potential for this type of model is higher than for an SVM model. From all system types, the highway systems have a higher accuracy score (DT: 63%, SVM: 66%, and ANN: 65%) than intersection systems (DT: 50%, SVM: 57%, and ANN: 58%). Next to the accuracy scores, the sensitivity of the models was tested. It was observed that the accuracy score does not significantly increase when the size of the data set is increased from 50 Service Orders to 75 and 100. What does influence the accuracy score is the detail of the input data. The more information that is present in the problem description, the higher the distinction between class labels. Another sensitivity of the model is the ability to correctly predict the class labels. A large difference is observed in the accuracy of each class label (electrical failure: 66-77%, software failure: 66-70%, mechanical failure: 9-36%, external damage: 18-52%, human error: 7-24%, and no problem: 0-15%). This difference is most likely caused by the high number of electrical and software failures with respect to the other labels. Therefore, there is more information available for these type of failures. The label *no problem* has a low accuracy because this label is by definition not a type of failure mode.

Given the evaluation of the model, we can conclude that the available data set of maintenance requests and solutions holds enough information on why systems fail to classify Service Orders based on types of failure modes. This increases the knowledge on failure behaviour of traffic control systems as well as the knowledge on the maintenance logs. We can also conclude that the ANN and SVM algorithms are most suitable to the data set and perform well in reference to similar studies from literature (Edwards et al., 2008; Wei et al., 2017). ANN has the highest improvement potential as the current algorithm is only a simple feed forward neural network. Multiple layered feed forward neural networks and recurrent neural networks should be more accurate. Even though, the accuracy scores are not high enough for direct implementation. The sensitivity analysis showed that the model has a large improvement potential on the quality and detail of the data set and the choice of class labels. Also, the model is limited on how features (i.e. informative words) are selected and the data set has been classified by the researcher, who has no experience with the maintenance activities. Based on these conclusions and limitations, the following steps are recommended before implementation:

- Assess alternative feature selection methods to decrease the number of uninformative features, creating a group of features with a higher distinctive value.
- Reassess the training and validation set on the aspects *quality and detail of the problem descriptions, size of the data set,* and *distinction of the class labels*.
- Assess alternative stemming algorithms for the Dutch language to increase the percentage of correctly stemmed words.

After implementation, it is recommended to extend the research to improve the performance of the model even more. For this phase, the following steps are recommended:

- Explore alternative text classification algorithms and rate their performance against the current model. As neural networks have the highest potential, both feed forward neural networks with multiple hidden layers and recurrent neural networks should be explored.
- Explore the possibility of combining the unstructured text data with structured and quantitative data for a more accurate or targeted classification method.

PREFACE

Dear reader,

Before you lies my thesis ”Predicting failure modes of traffic control systems by classifying unstructured data of maintenance requests”. It has been written to complete the graduation requirements of the Industrial Engineering & Management program at the University of Twente. The assignment was executed at the request of Dynniq Nederland B.V. from February 2020 to July 2020.

I am very grateful for the opportunity I got at Dynniq to show where my skills lie and to learn about real life maintenance optimization issues. I especially want to thank Chris Heineman and Jeroen Kuip for creating the thesis opportunity and providing me with all the information I needed to execute my research at Dynniq. Even though my time in Amersfoort has been minimal due to the coronavirus, I have become to like the company and the atmosphere at the CS department.

I also want to thank my supervisors from the University of Twente, Matthieu van der Heijden and Asad Abdi, for guiding me through the graduation assignment and providing me with the required feedback to write a proper thesis. Your keen eye for detail and text structuring has brought my thesis to a higher level.

Finally, I want to thank my family and friends, who always supported me in my ambitions to get more than just a diploma out of my studying years. With amazing adventures during the honours programme and the World Solar Challenge in Australia, I have learned more than I could have ever hoped. I take these experiences with me in my road towards a great career.

All that remains for me is to wish you a pleasant read.

Robin Haandrikman

Deventer, August 11, 2020

CONTENTS

List of Figures

List of Tables

List of Definitions

Definition	Description
Accuracy score	Percentage of correctly classified instances.
Cross-Industry Standard for Data Mining (CRISP-DM)	Methodology for structuring data mining projects.
Data mining	Extracting information from data and transforming these data into a model.
Document vector	A document or sentence in the mathematical form of vector.
Failure mode	Reaching such a state that the intended function of the part or system can no longer be fulfilled.
Feature selection	The selection of words or groups of words, also known as terms or features.
Feed forward neural network	A neural network that only performs calculations from left (the input) to right (the output).
Gain Ratio	Feature selection method that selects features with the highest informative value.
Gini Index	Feature selection method that selects features based on the chance of misclassification.
Kernel function	Method used in a Support Vector Machine for solving a non-linear problem using linear classifiers.
Part-Of-Speech tagging	Providing words with a tag that explains the type of word (e.g. noun, verb, or adverb).
Recurrent neural network	A neural network that performs calculations from left (the input) to right (the output) and has the option to repeat or improve a calculation in the hidden layer.
Term weight	A calculated number that is linked to a term (i.e. word or group of words). For example the occurrence of a word in a sentence.
Text mining	Extracting information from text fields by transforming words to a numerical form like occurrence of words in a text.
Traffic Control System	System next to the road that controls traffic flow via external systems like traffic lights and information signs.
Unstructured data	Textual data with a free form. Often used as a free text field like *description*.

List of Abbreviations

Abbreviation	Definition
ANN	Artificial Neural Network
BoW	Bag of Words
CRISP-DM	Cross-Industry Standard Process for Data Mining
CS	Customer Service
DT	Decision Tree
EC	EuroController
FME(C)A	Failure Mode Effects and Criticality Analysis
IF	Intermediate Form
ITIL	Information Technology Infrastructure Library
MDL	Minimal Description Length
NLP	Natural Language Processing
OBJ	Object
OOP	Object Oriented Programming
OS	OutStation
PMML	Personalized Print Markup Language
POS	Part-Of-Speech
QP	Quadratic Programming
RCA	Root Cause Analysis
RNN	Recurring Neural Network
SCN	Service Contract Number
SLC	Service Location Code
SMO	Sequential Minimal Optimization
SO	Service Order
SOC	Service Order Corrective
SOP	Service Order Preventive
SVM	Support Vector Machine
TF$\times$IDF	Term Frequency $\times$ Inverse Document Frequency
VRA	Verkeersregelautomaat (EN: Traffic Light Controller)
VRI	Verkeersregelinstallatie (EN: Traffic Lights)
VSM	Vector Space Model
WKS	Wegkantstation (EN: Roadside Station)
WO	Work Order

1. INTRODUCTION

We start this thesis by introducing the background of this research. In section 1.1, we will shortly introduce the company. Then, we continue with the problem description in section 1.2 and the research questions in section 1.3. Next, we discuss the methodology that is used to guide the research in section 1.4 and we end the chapter in section 1.6 with the outline of the report.

1.1. COMPANY BACKGROUND

Dynniq helps to manage energy and mobility flows in society through advanced technological solutions. The mission is to enable people, data and goods to reach their destinations safely, sustainably, and efficiently. As flow increases, the technology that guides this flow must increase as well. One way of guiding the flow is the use of traffic control systems. These systems must be maintained to ensure continuity of flow. Dynniq provides this maintenance through service contracts. The strategy for maintenance is based on preventive and corrective maintenance. Preventive maintenance is done every one or two years, depending on the service contract. Corrective maintenance is done when a failure is reported. Depending on the service contract, the response time is two, four or eight hours. This maintenance strategy is found to be both conservative and ineffective. This can be concluded from a high ratio (56%) of corrective maintenance actions with regard to preventive maintenance. Also, one general checklist is used as a protocol for all preventive maintenance activities. Improving the maintenance strategy is key as the service contracts are becoming less valuable due to a lower selling price of the traffic control systems.

One way of improving the maintenance strategy is by analysing logged data from maintenance activities on the traffic control systems. These data include information such as the repair actions that are executed, the location of the systems that are repaired, the date and time corresponding to the repair and product information corresponding with the repaired system. Analysing the data would increase the knowledge on the traffic control systems. This knowledge will be used to improve the maintenance strategy.

1.2. PROBLEM IDENTIFICATION

As Dynniq sees the need to decrease their maintenance costs due to an expected decrease in revenue, efficient planning in maintenance is necessary. This can either be done by improving the planning method, which lowers the required capacity and resources needed, or by improving the execution of maintenance, which can result in less corrective maintenance activities or the removal of ineffective preventive maintenance activities. In both cases, the workload will decrease which leads to a decrease in maintenance costs. In this research, we focus on improving the maintenance strategy. Dynniq sees this as the best option to decrease maintenance costs. To improve the maintenance strategy, we will look at the failure behaviour of the traffic control systems. If failure behaviours are known, accurate maintenance actions can be linked to the traffic control systems. This has an influence on the design of maintenance actions as well as the use of capacity like labour and materials.

The outcome of the research must be possible to implement in the current maintenance strategy. Therefore, a set of guidelines for improvement of the maintenance strategy is desired in the form of an implementation plan. These guidelines can be used to improve the effectiveness of preventive maintenance actions and increase the ratio of preventive maintenance actions with respect to corrective maintenance. Such guidelines can be created with a model that uses the available data from maintenance activities. The model analyses

the available data and provides information such as why a system has failed. The model must be available within the company for future assessments and improvements. A new model increases the probability of making the maintenance strategy more sustainable as it provides insight into the available data. As a result, the maintenance plan will be more reliable and resources can be used more efficiently.

1.2.1. CORE PROBLEM

From this situation, we can deduce the core problem. The core problem was given by experts from the Customer Service Operations department, but has been analysed by the researcher. We can find the core problem in the problem cluster in Figure 1.1: "The current maintenance strategy is expected not to be financially future proof". The problem cluster also shows us that the scope of the assignment only partially solves the core problem. With a data analysis, knowledge on maintenance can be increased but the value of service contracts will still decrease. Solving the latter is out of scope of this assignment.

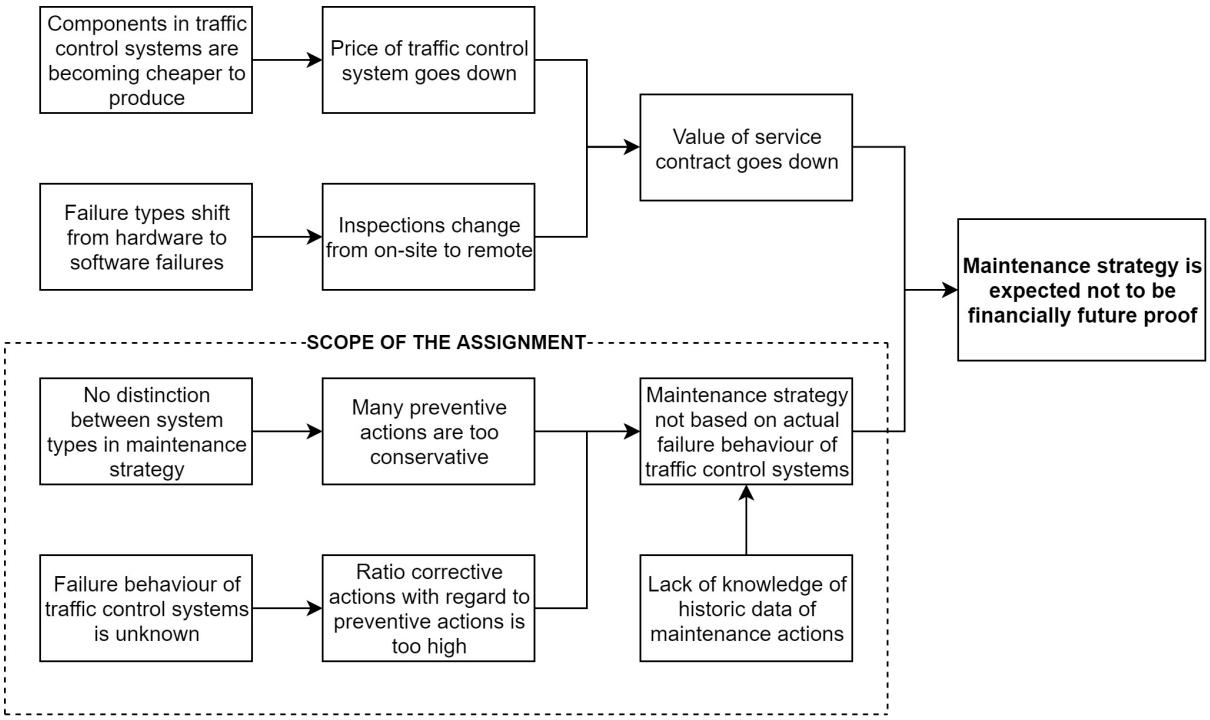


Figure 1.1: Problem cluster

1.2.2. SCOPE

The research focusses on an improvement solution based on available data from maintenance requests and solutions. Only the data on highway systems and intersection systems will be analysed as these systems are produced and maintained by Dynniq. The maintenance types are reduced to preventive and corrective maintenance. Other types of maintenance are executed during the installation of a traffic control system, which is out of the research scope. Methods from literature will be combined or adapted to solve the research problem. Improvements regarding planning optimizations are out of scope as well as all financial aspects. The results of the assignment will be collected in an implementation plan. The execution of the implementation plan is out of scope.

1.3. RESEARCH QUESTIONS

Based on the problem cluster, we can outline the research goal, which is written as the main research question. To answer this question, we must answer several sub-questions which each comply with a chapter in this report. Taking the core problem and the scope of the assignment into account, the main research question will be:

"How can historical data on maintenance requests and solutions improve the effectiveness of any kind of maintenance, making it more future-proof?"

We start the research with an analysis of the company, its maintenance strategy and the data logs they keep. The logs hold information about why systems have failed and how they are repaired. The analysis ends with the outline of the improvement potential to which the research is aimed. In chapter 2 we will answer the first research question.

I. *"What is the current maintenance strategy and what is the improvement potential?"*
   a. *"What is the definition of the current maintenance strategy?"*
   b. *"What improvements of the maintenance strategy are desired?"*
   c. *"What information does the gathered data hold?"*
   d. *"What is the company's performance based on the data?"*

The analysis of the business and the data set will provide us with an improvement potential. To make use of this potential, a literature review will be executed to find theoretical foundation and techniques for modelling towards such potential. In chapter 3 we will answer the second research question.

II. *"What methods does literature provide for maintenance strategies based on data about maintenance requests and solutions?"*
   a. *"What maintenance optimization methods are available for traffic control systems?"*
   b. *"What maintenance optimization methods are based on data of maintenance requests and solutions?"*
   c. *"What methods can increase the knowledge about the failure behaviour of the traffic control systems?"*
   d. *"Which parts of the research problem can and cannot be solved with the available literature?"*

With the methods from literature known, we can design a model which will provide answers to the core problem. To tailor the company's needs, we must combine methods from literature with the available data set. In chapter 4 we will answer the third research question.

III. *"How can data be modelled to provide guidelines for optimizing the maintenance strategy?"*
   a. *"What techniques can be used to improve the current maintenance strategy?"*
   b. *"What changes to or combinations of models are chosen as the final model?"*
   c. *"How can results be summarized in clear and useful guidelines?"*

After building the model, we will evaluate the results. The evaluation includes a conclusion on whether the created model is suitable for solving the main research question. In chapter 5 we will answer the fourth research question.

IV. *"How can the results from the data model be used to improve the maintenance strategy?"*

   a. *"What results does the data model provide?"*
   b. *"What impact do the results have on the research problem?"*

Finally, we will discuss the implementation of the model in the daily work routine. In chapter 6 we will answer the fifth research question.

V. *"How can the data model be implemented in the maintenance strategy?"*

   a. *"What changes have to be made to implement the data model?"*
   b. *"What impact do the changes have on the stakeholders?"*
   c. *"Which steps have to be taken to complete the implementation?"*

1.4. METHODOLOGY

We chose the Cross-Industry Standard for Data Mining (CRISP-DM) methodology to structure the research. CRISP-DM is the most often used methodology for data mining projects, according to polls from 2007 and 2014 (Piatetsky, 2014). As Larose (2005) states, the methodology must be seen as an entire process, from communication of the business problem through data collection and management, data preprocessing, model building, model evaluation, and finally, model deployment. Even though it is not being updated for the new 'Big Data' challenges (Piatetsky, 2014), the methodology is often used as a blueprint for conducting any data mining or data analysis project (Shearer, 2000). It is built for researches based on historical data and it is assumed that the reader has little knowledge about the business objectives and available data (Larose, 2005; Shearer, 2000).

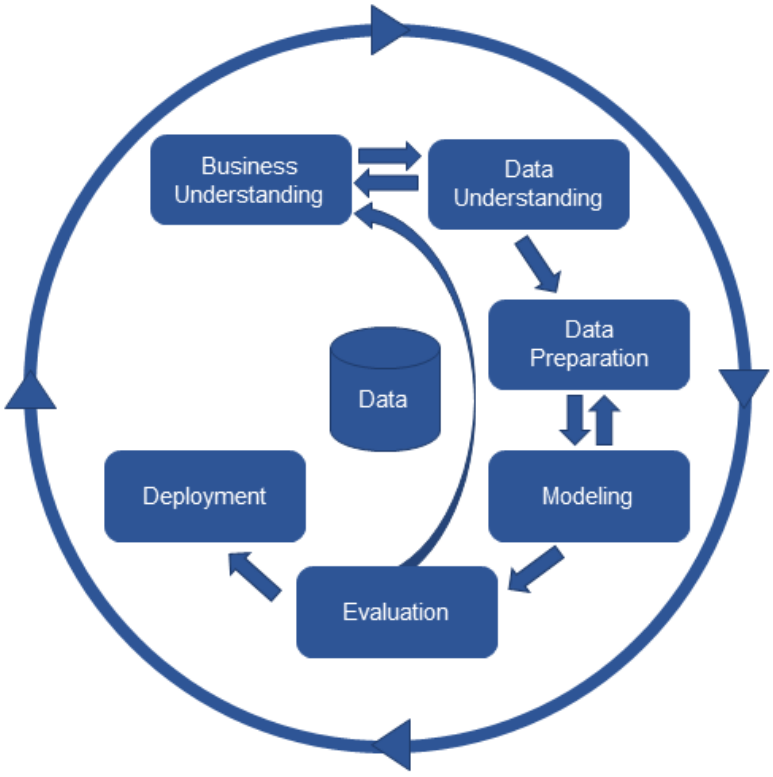


Figure 1.2: Visual representation of the CRISP-DM methodology

We chose this methodology because the starting point and goal of the research comply with the starting point and goal of the methodology. Its phases (Figure 1.2) guide the execution of the research. The six phases, *business understanding*, *data understanding*, *data preparation*, *modelling*, *evaluation*, and *deployment*, are directly applicable to the research questions that need answering (Table 1.1). The phases have no static order as data mining is an iterative process. Shearer (2000) provides clear tasks and outputs for each phase. We have extended the methodology with an extensive literature review on data preparation and modelling techniques during the data preparation and modelling phases.

Table 1.1: CRISP-DM phases and corresponding research questions

CRISP-DM phase	**Research questions**
Business Understanding	I. "What is the current maintenance strategy and what is the improvement potential?"
Data Understanding	I. "What is the current maintenance strategy and what is the improvement potential?"
Data Preparation	II. "What methods does literature provide for maintenance strategies based on data about maintenance requests and solutions?"
Modelling	II. "What methods does literature provide for maintenance strategies based on data about maintenance requests and solutions?" III. "How can data be modelled to provide guidelines for optimizing the maintenance strategy?"
Evaluation	IV. "How can the results from the data model be used to improve the maintenance strategy?"
Deployment	V. "How can the data model be implemented in the maintenance strategy?"

1.5. Contribution of the research

The contribution of this research lies in the type of classification model that is designed. Many articles are available on the subject of text classification, but less for the Dutch language and none for maintenance activities on traffic control systems. This research contributes with a method for the classification of Dutch text data based on failure modes of traffic control systems.

1.6. Outline

The remainder of this research starts in chapter 2 with an introduction of the company, its maintenance strategy, and the available data set on maintenance requests and solutions. This covers the first two phases of CRISP-DM, *business understanding* and *data understanding*. The chapter also provides a performance analysis, which resulted in an improvement potential. Chapter 3 covers the theoretical background that is required to address the improvement potential properly. As mentioned in section 1.4, this literature review is an extension to CRISP-DM. Next, the theoretical background is used in chapter 4 to design a model that transforms the available data set into an informative form that can be used to answer the main research question. This covers the *data preparation* and *modelling* phases of CRISP-DM. The next phase is the *evaluation* phase, which is covered in chapter 5. Here, the results of the model and the model's limitations are evaluated. Then, chapter 6 presents the required and recommended steps for implementation of the model, which complies with the final step of CRISP-DM, *deployment*. Finally, chapter 7 covers the conclusions, limitations, and recommendations.

2. Context Analysis

In this chapter we will discuss the analysis of the business and the available data set. With this analysis, we will answer the first research question: *"What is the current maintenance strategy and what is the improvement potential?"*. In section 2.1, we will discuss the Customer Service Operations department that provides maintenance solutions. Next, we will discuss the traffic control systems that the company produces and the strategy for maintenance on these systems. Then, in section 2.2 we go into the definition of the available data before we discuss the company's performance in section 2.3. Based on the performance, we will discuss the improvement potential in section 2.4 and end the chapter with conclusions in section 2.5.

2.1. Business Understanding

To understand the business fully, we will discuss the analysis of the company, the traffic control systems, and the current maintenance strategy. This knowledge is required to fully understand the origin of the research problem.

2.1.1. Organization

Dynniq Group has three separate divisions: Energy, Parking, and Mobility. The Energy division is specialised in the design, delivery, and maintenance of electrical and gas infrastructure, Parking is specialised in innovative parking solutions and Mobility, the main division, is specialised in a safe and efficient flow of traffic. Dynniq Mobility is run from the main office at Amersfoort, where Dynniq Mobility NL is located. One of the departments within Dynniq Mobility NL is known as Customer Service Operations (CS Operations). This department is responsible for maintenance on the traffic control systems.

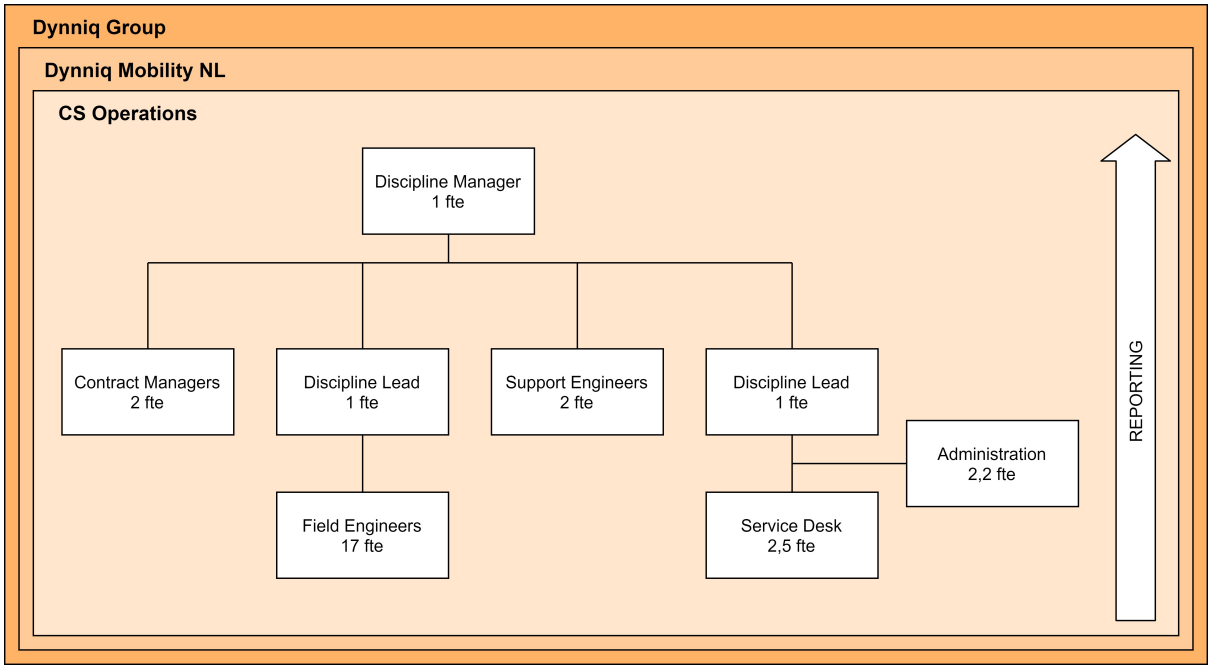


Figure 2.1: Organization chart CS Operations

CS Operations' organization chart (Figure 2.1) consists of Contract Managers, Support Engineers, Field Engineers, Service Desk, and Administration. The Field Engineers and Service Desk both have a discipline lead. The Contract Manager is responsible for ensuring that the service contracts are respected, the Field Engineers execute the maintenance on the traffic control systems as communicated in the service contracts, the Support Engineer helps

the Field Engineers with specialist technical support, the Service Desk handles the first line orders for corrective maintenance work, and Administration draws up the service contracts and handles the maintenance reports. The hierarchy shown in Figure 2.1 shows to who a person in the chart must report, the first person upstream. The department has short lines for daily communication to enable employees to execute their work properly.

2.1.2. Traffic Control Systems

The systems for which maintenance is required are either produced by Dynniq or third parties. As mentioned in section 1.2.2, in this research we will focus on the systems produced by Dynniq. These systems control traffic flows at intersections and information signs along highways. The intersection systems are known as 'verkeersregelautomaat' (VRA), which is the dutch term for traffic light controller. These systems have a system code starting with EC, which stand for EuroController. The highway systems are known as 'wegkantstation' (WKS), which is the dutch term for roadside station. These systems have a system code starting with OS, which stand for OutStation. For ease of understanding, we will refer to the systems as intersection system and highway system. Table 2.1 shows the current number of systems active to date. All systems are constructed such that maintenance can be done quickly. Electrical modules are used to control a certain feature of the system, for example remote access. When a problem occurs with the remote access, the full module can be replaced. Another benefit of the modular system is the option to modify on customer demand before installation.

The intersection systems control the flow of traffic across an intersection. Dependent systems are traffic lights and detection loops. Detection loops are electrical wires that run through the top layer of the road. When the magnetic field between the wires is interrupted by the presence of a metal object (e.g. car, bike or truck), the traffic control systems know that a vehicle is present. The more recent systems are interconnected to ensure possibilities like a green flow throughout a city.

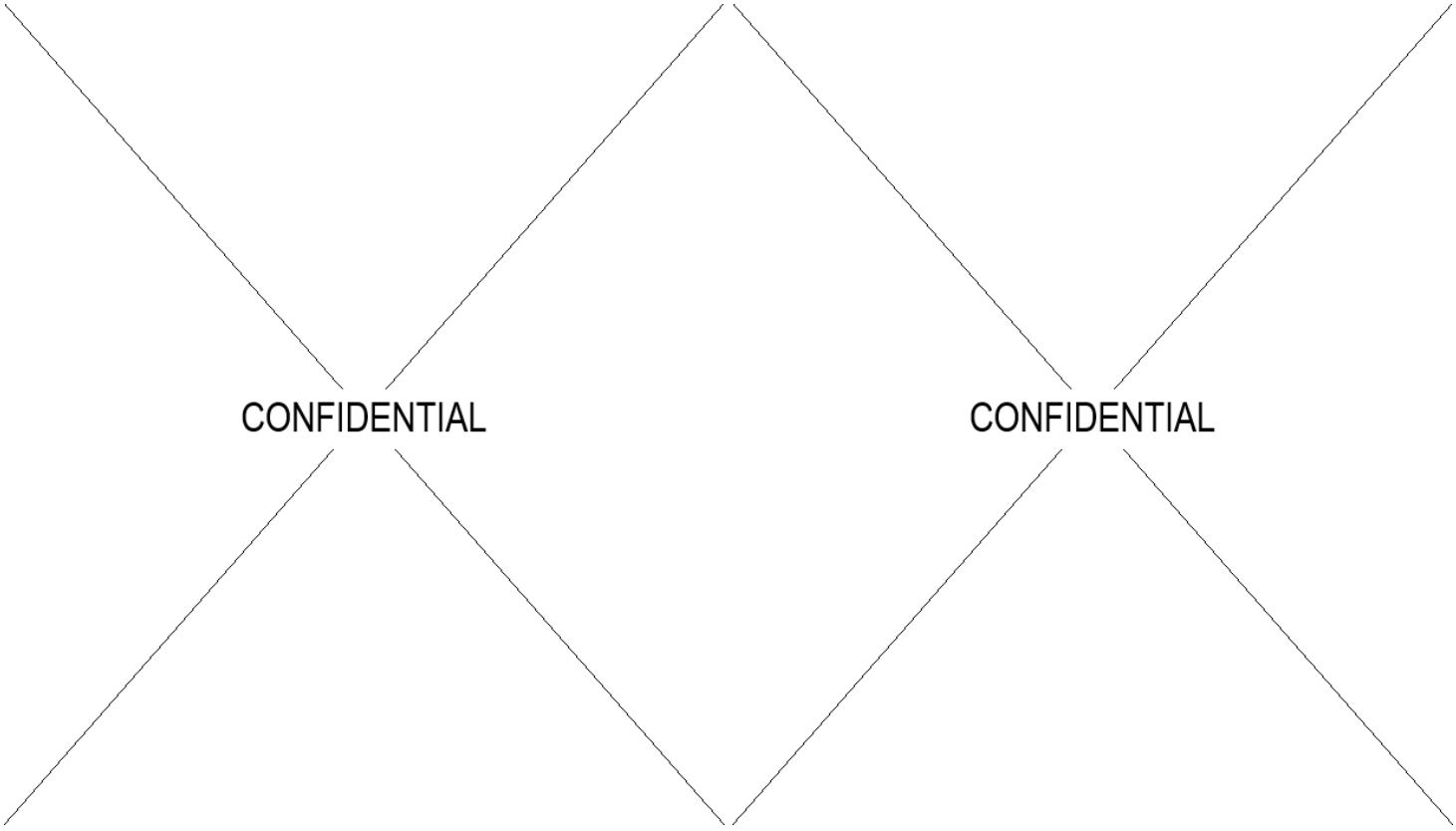
The highway systems control the flow of traffic along a highway, which is mainly done through information signs. Examples of information are traffic jams, rush hour lanes, and speed limitations. There is a high number of systems that are interconnected due to the long stretch of road.

Table 2.1: Traffic Control Systems currently active

Highway System		**Intersection System**	
System code	**Active number**	**System code**	**Active number**
EC-1	*Confidential*	OS-0	*Confidential*
EC-1 iVRI	*Confidential*	OS-2	*Confidential*
EC-2	*Confidential*	OS-3c	*Confidential*
EC-2 Compact	*Confidential*	OS-3	*Confidential*
EC-2 iVRI	*Confidential*	OS-3/4	*Confidential*
EC-2.5 iVRI	*Confidential*	OS-4	*Confidential*
EC-2L	*Confidential*	OS-5	*Confidential*
EC-200D	*Confidential*	OS-6	*Confidential*
EC-2.5	*Confidential*		
Total	*Confidential*	**Total**	*Confidential*

2.1.3. MAINTENANCE STRATEGY

The maintenance strategy at Dynniq is based on the Information Technology Infrastructure Library (ITIL). ITIL is a systematic approach to the delivery of quality IT services (van Bon, 2011). It describes the different stages of the service lifecycle: strategy, design, transition, operation, and continual improvement. In the operational stage, the stage in which the maintenance activities are executed at Dynniq, ITIL describes the workflow of any occurring maintenance event. The maintenance activities at Dynniq are split into two main groups: preventive maintenance (Figure 2.2a) and corrective maintenance (Figure 2.2b).



(a) Preventive maintenance (b) Corrective maintenance

Figure 2.2: Maintenance flow based on ITIL 2011

Preventive maintenance

Preventive maintenance at Dynniq can be defined as a full check of a traffic control system. A protocol is used which describes which maintenance tasks must be executed. The protocol has two versions, a full check called 'standard' and a partial check called 'basic'. The standard check includes the operation of a traffic control system and the communication with dependent systems like traffic lights, detection loops and information signs. This type of maintenance can be done for up to *confidential* systems per day per Field Engineer. The basic check, which can be done up to *confidential* times per day per Field Engineer, excludes all checks regarding *confidential*. When the loops, which lie in the top layer of the road, are not functioning properly, the system will show an error message.

Preventive maintenance is executed once every one or two years, depending on the customer demand. This and the execution of the maintenance, as can be seen in Figure 2.2a is prescribed in a service contract. A Senior Field Engineer creates the maintenance planning two or three months in advance. Inputs for the planning, like due dates and contractual agreements, come from the Contract Managers. The planning is then communicated to the customer. When the date is set, a new Service Order Preventive (SOP) is created in the

Enterprise Resource Planning (ERP) system. An ERP system is a tool used for digital support of company processes. All information regarding the maintenance activity will be added to the created SOP. The system provides a work order which prescribes the maintenance activities to be done. The responsible Field Engineer executes the maintenance activity and reports back via a maintenance report. When the activity is completed, details from the maintenance report are added to the SOP, the customer is informed and the order is closed.

Preventive maintenance is mainly planned for traffic control systems at intersections. *The execution of preventive maintenance is confidential, but known to the researcher.*

Corrective maintenance
Corrective maintenance at Dynniq can be defined as a reaction to a trigger within a response time as prescribed in a service contract. This reaction is initiated by the Service Desk. As can be seen in Figure 2.2b, the trigger is either a traffic control system that shows an error message or a call received from a customer. Based on the information from the trigger, the Service Desk discusses the problem with a Field Engineer. When the problem solving method is not clear, a Support Engineer is requested for technical support. If an action is required, a Service Order Corrective (SOC) is created in the ERP system. The system reports back what requirements for maintenance are in the service contract. For instance, the response time can be two, four or eight hours. When the Field Engineer has executed the maintenance, he reports on the cause and solution of the problem. The Service Desk processes this information and either finishes the Service Order or plans a follow up order. When the maintenance activity is completed, the Service Desk reports back to the customer. If the response time is not met, a penalty is applied.

Alternative maintenance actions
Next to preventive and corrective maintenance, one component is maintained by a different policy. The highways systems have a backup power source in the form of batteries. *The battery maintenance strategy is confidential, but known to the researcher.* When the batteries are replaced, the system backup ability will be seen as 'as good as new'. This is currently the only type of maintenance that differs from the standard preventive and corrective workflows.

2.2. DATA UNDERSTANDING

Now that we understand the business, we will analyse what information is gathered in the available data set. To do so, we briefly discuss the origin of the data set and then describe the information that the data set holds.

2.2.1. COLLECTION OF DATA

The available data set is a combination of extractions from the ERP system. The data represents the logs of maintenance requests and solutions. New information is added by employees from the Service Desk, who handle new and active maintenance activities, also known as Service Orders (SO). The ERP system can be accessed by all personnel from the CS Operations department. Dynniq is collecting data since 2010. In 2017, a new version of the ERP system was put into operation. For this research, we will use all dates from 2011 to 2019. From 2010 and 2020 there is no full year of data available and we will therefore exclude these data.

2.2.2. Description of data

To show what information is present in the data set, we must perform a performance analysis. Before we go into the analysis, we will discuss the description of the most important attributes in this subsection. The attributes are grouped and the groups are explained. In this phase of the research, we have excluded undesired attributes from the data set.

Object

A new database entry is started with the selection of the traffic control system that must be maintained. Such a system is also known as an object. Information about the object is collected in the Object List, one of the extractions from the ERP system. The Object List contains the unique object code (e.g. OBJ170000002), the service location code (e.g. SLC000001), an address or mile marker code (e.g. A009 058.4), and the service contract number in which the maintenance requirements are collected (e.g. SCN70100038).

Service Order

When a new Service Order is created, the database entry automatically gets a unique code which represents a maintenance request, for example 'SOC17018000001'. All other attributes that are relevant to the maintenance request, such as contract and customer information, timestamps, and maintenance reports are linked to this code. The code provides information on the maintenance type (e.g. SOC: Service Order Corrective), the department code (e.g. 170), the year which the entry is created (e.g. 2018), and the yearly entry number (Figure 2.3).

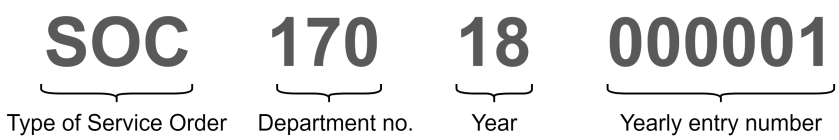


Figure 2.3: Service Order Number

Location

Each object has one location, but one location can house multiple objects. Therefore, the location has a unique code as well (e.g. SLC000001). Relevant information about the location is linked to this code, such as an address or mile marker code and a description of the location. A location description shows what type of objects are available at the location.

Problem description, cause, and solution

Three attributes, the problem description, cause, and solution, are directly related to the maintenance job. These attributes are text fields and entered by the Service Desk (description) or the Field Engineer (cause and solution). The text fields describe what the customer observed, how the problem was caused, and how it was solved by the Field Engineer.

Reference point

The changes in the maintenance logs are recorded with reference points. These points are date- and timestamps. A part of the points are entered manually, which represent the following moments in time: *maintenance request received, order entered in ERP system, start and end of the maintenance job,* and *order closed*. The other reference points are automatic entries, which are connected to the order status. This can be *Open, In Process, Technically Finished, Administratively Finished,* and *Cancelled*. When the status changes, the system logs the time.

Spare part

Information on replacement parts is collected in the Service- and Maintenance Entry List, another extraction from The ERP system. This file collects per Service Order which costs are linked to what Service Order. This include the 12NC number of a replacement part (e.g. N9586 149 44200).

2.3. Company Performance

With the business and data set understood, we can continue with the current performance of the company. The performance is based on an analysis of the available data set. As we have defined in the problem cluster (section 1.2), there is a lack of knowledge on how maintenance is executed and what information is in the maintenance logs. Moreover, the current strategy is not based on actual failure behaviour of the traffic control systems. In this section we will discuss the performance parameters that provide insight in what part of the maintenance strategy lacks performance. From the problem cluster, we can deduce a lack of performance to a lack of knowledge, which therefore must be further researched.

The performance analysis is executed to find an improvement potential which we can focus on in our literature review in chapter 3. Doing this will direct the problem solving method towards this potential. We will start with the discussion on a high level, only considering corrective and preventive maintenance. Next, we will discuss the Service Orders which could not be finished in one attempt (i.e. one visit to the traffic control system). After that, we will compare the intersection and highway systems. Finally we discuss the spare parts and make a comparison between the system groups and maintenance types based on the spare parts.

Preventive and corrective maintenance

As mentioned in section 2.1.3, the company's maintenance strategy is based on preventive and corrective maintenance. As preventive maintenance is part of a service contract, it is in general a profitable maintenance activity. Corrective maintenance on the other hand comes with penalty fees when contractual agreements are not fulfilled. Therefore, the focus is to keep all systems running while performing as few corrective actions as possible. Figure 2.4 shows us the number of Service Orders per object per year for corrective (SOC) and preventive (SOP) maintenance. The number of preventive orders per object has grown from *confidential* to *confidential* order per object per year. An increase of preventive maintenance actions should decrease the number of corrective maintenance actions. However, the number of corrective orders has grown slightly as well. When we compare the maintenance types as a ratio of all maintenance activities, an increase of the ratio of preventive maintenance activities can be observed (see Figure 2.5).

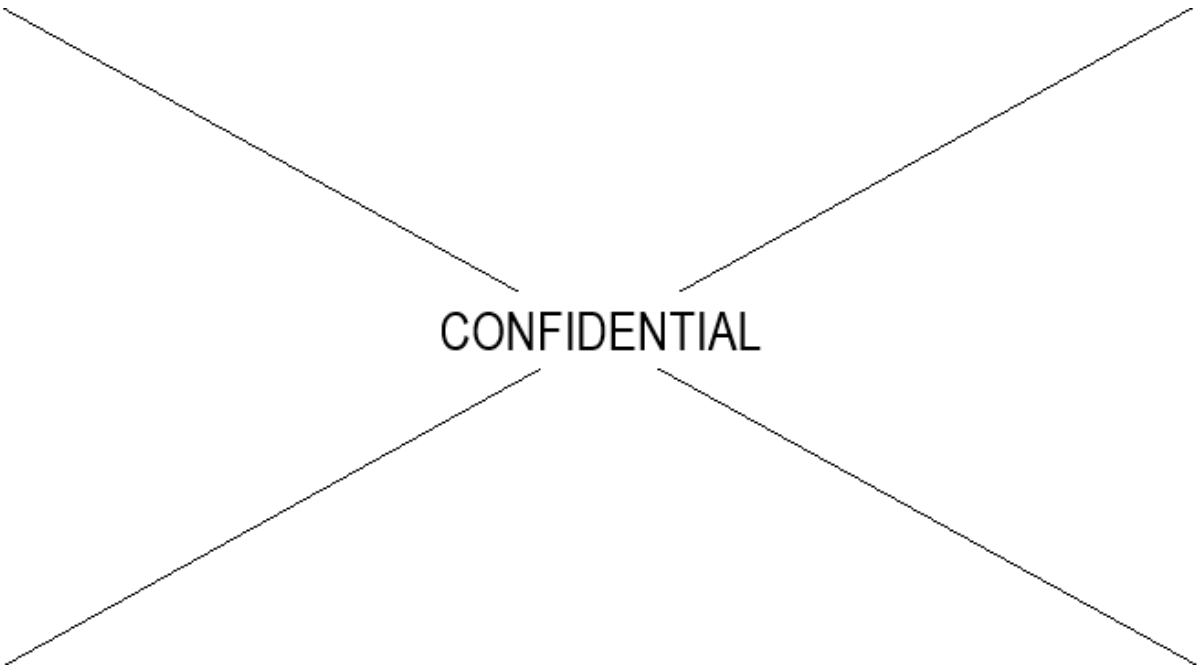


Figure 2.4: Number of service orders per object per year

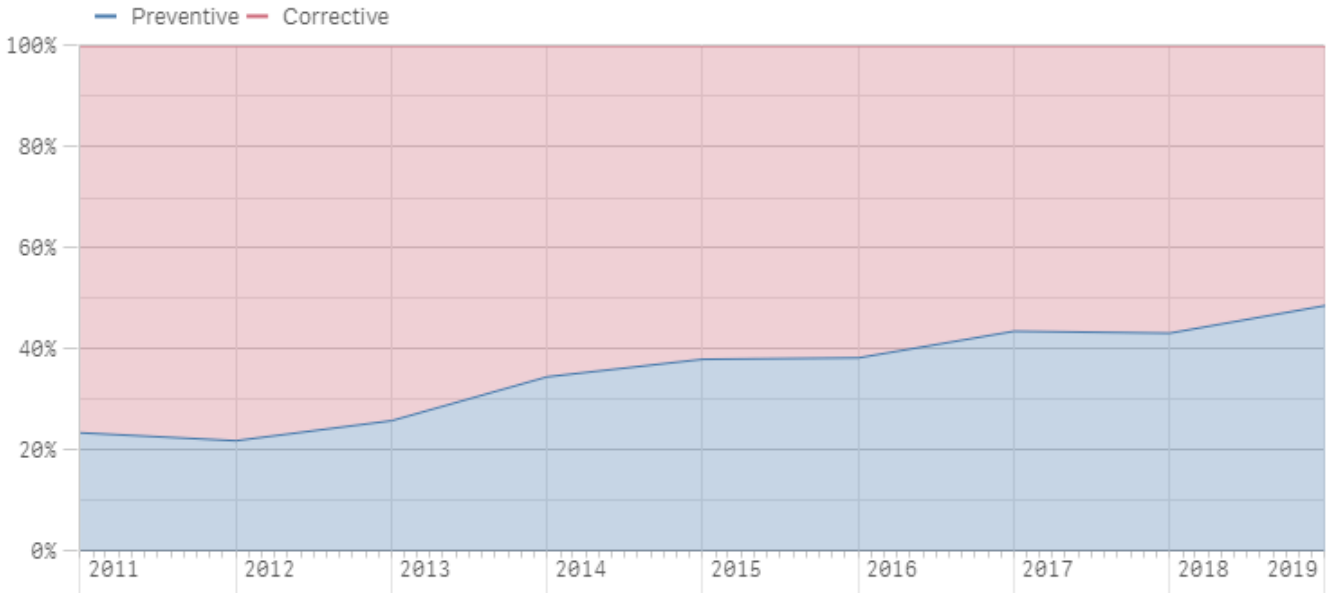


Figure 2.5: Ratio corrective vs. preventive maintenance activities per year

Completion of Service Orders
A part of the Service Orders cannot be solved during the first attempt, for example due to the lack of a good problem solving approach or limited resources. When that happens, a second visit will be planned. *The performance of Figure 2.6 is confidential, but known to the researcher.*

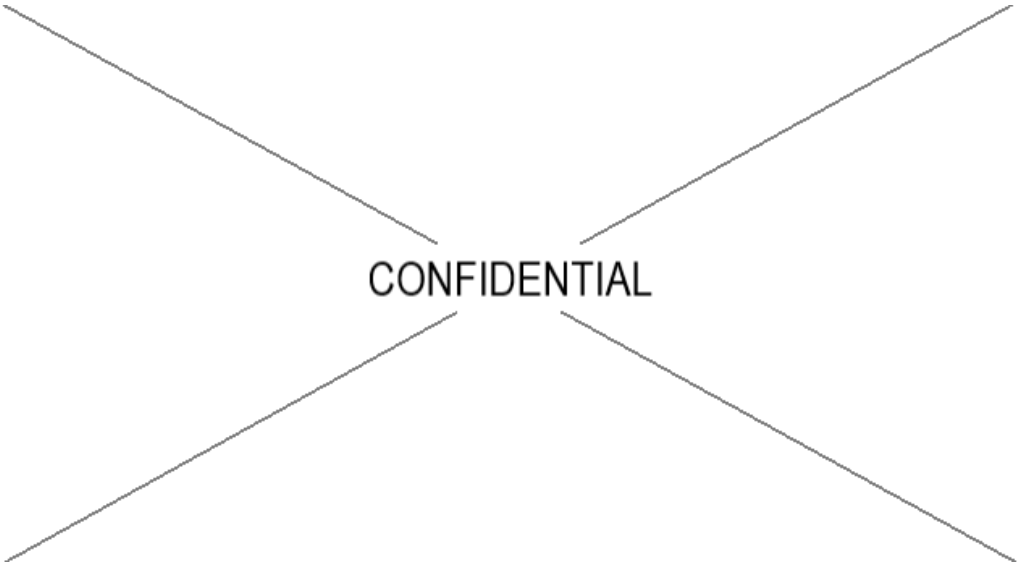
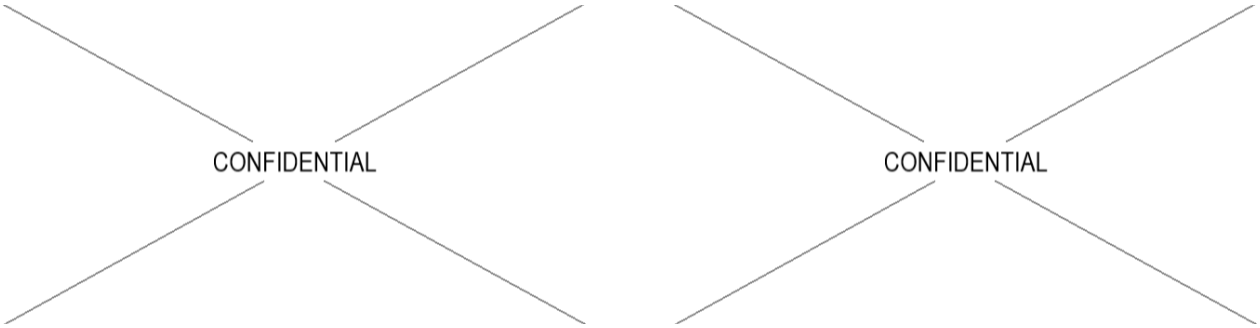


Figure 2.6: Percentage of orders that need two or more visits

The performance of Figures 2.7a and 2.7b is confidential, but known to the researcher.



(a) Order with two attempts (b) Orders with three or more attempts

Figure 2.7: Percentage of orders that need two or more visits - extended

System groups

Zooming into the system groups (i.e. highway and intersection systems), we see a large difference when we compare corrective and preventive maintenance as a ratio to all maintenance activities. In Figure 2.8 we can observe that preventive maintenance for highway systems is only introduced since 2018, with the exception of 26 activities in 2012. The reason behind the lack of preventive maintenance lies, according to a Senior Field Engineer, in the simplicity of older highway systems. Preventive maintenance has therefore been chosen to be ineffective for these older systems. In Figure 2.9 we observe that for intersection systems, the ratio has been roughly evenly divided since 2014.

When we look at the number of service orders per object per year (Figure 2.10a & 2.10b), we can observe an increasing trend for both corrective and preventive maintenance activities on intersection systems. The number of maintenance activities for highway systems remained roughly unchanged, with the exception of the introduction of preventive maintenance in 2018.

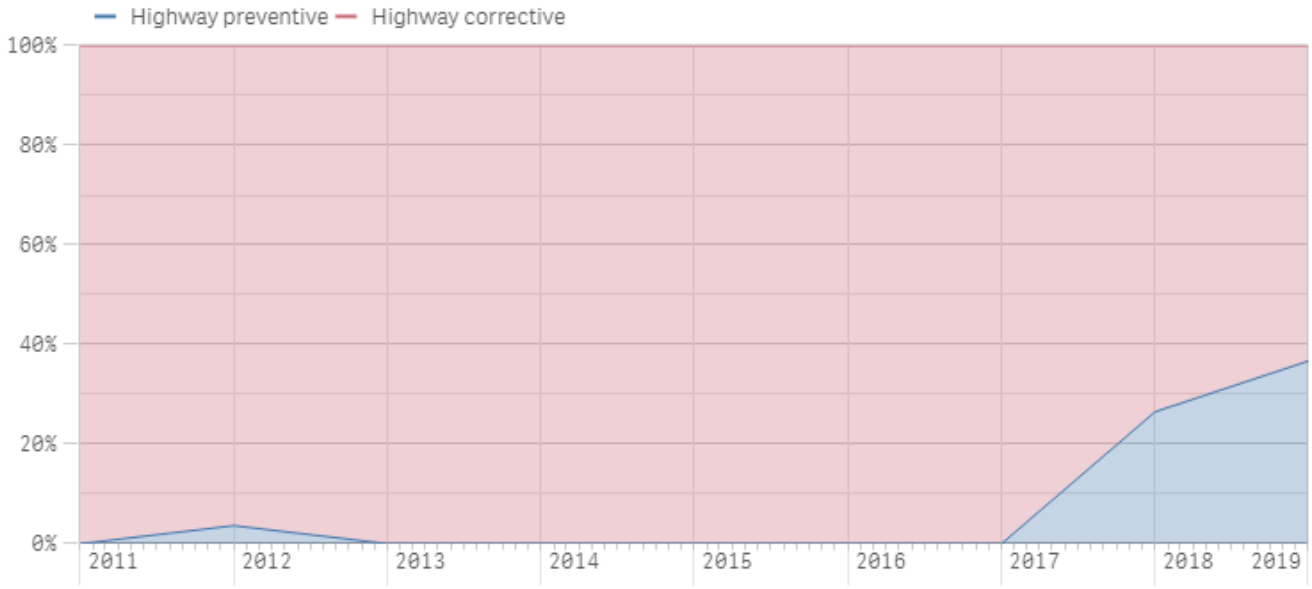


Figure 2.8: Ratio corrective vs. preventive maintenance activities per year for highway systems

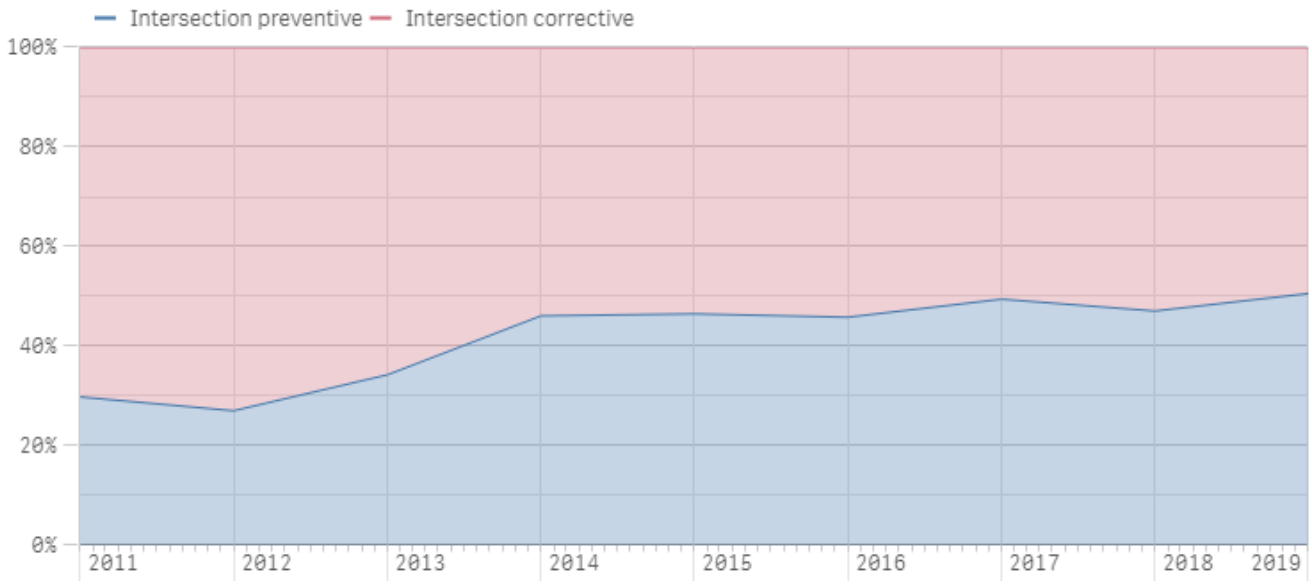
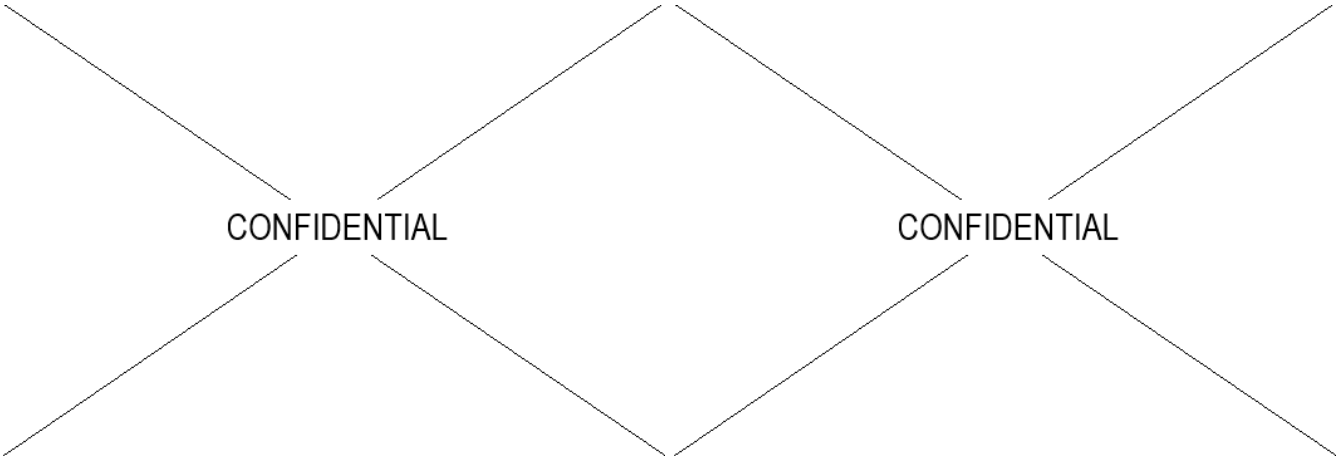


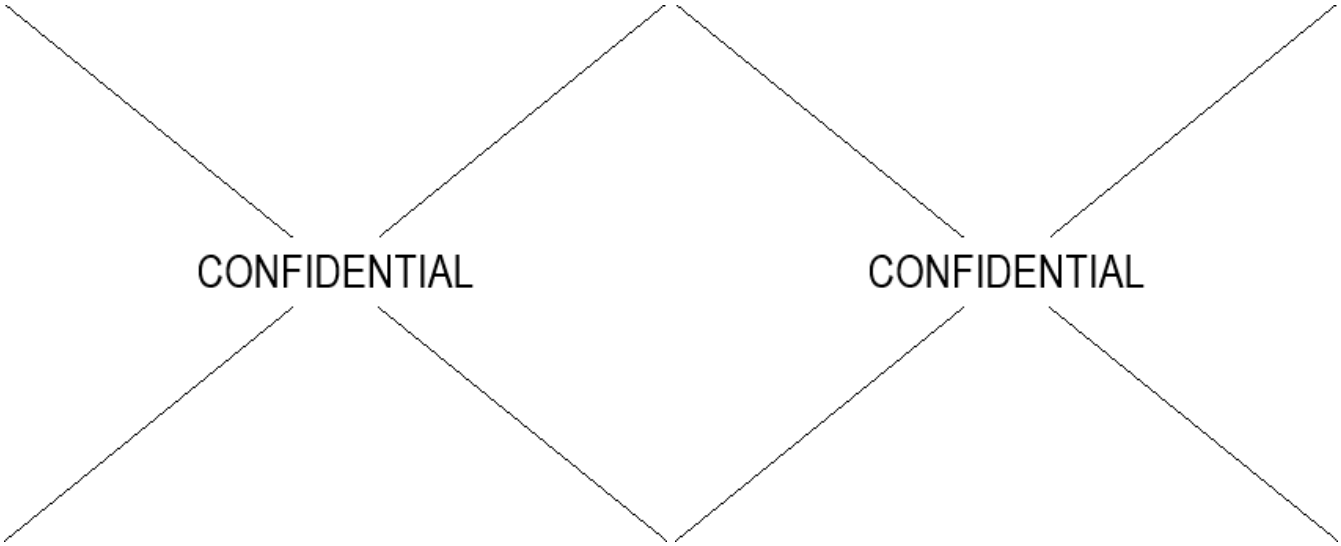
Figure 2.9: Ratio corrective vs. preventive maintenance activities per year for intersection systems



(a) Highway systems (b) Intersection systems

Figure 2.10: Number of service orders per object per year per system group

Replacement parts
When we consider the replacement parts that were needed in the last nine years, we can observe in Figure 2.11a an increasing trend for both highway and intersection systems. From Figure 2.11b we can observe that almost all replacements have happened during corrective maintenance. The total number of replacements have more than doubled since 2011, even though the number of preventive maintenance orders has increased over the years.



(a) System group (b) Service Order type

Figure 2.11: Number of replacement parts per object per year

2.4. Improvement Potential

In section 2.3, we discussed several performance parameters related to the problem cluster. These parameters represent how maintenance on traffic control systems has been executed since 2011 and what potential for improvement is available. In this section, we will identify this improvement potential, which we will use as a base for our literature review in chapter 3. Then, we can direct the problem solving method towards an approach that complies with the improvement potential.

The performance shows several potential improvements. First of all, we have observed an increase in preventive maintenance activities in Figures 2.4 and 2.5. Figures 2.10a and 2.10b show that the increase is present for both highway and intersection systems. As preventive maintenance is a method to reduce overall maintenance costs, the number of corrective maintenance actions per object should decrease to ensure profitability. However, for the intersection systems we observe a slight increase in corrective maintenance actions. This implies a minimal effectiveness or even a redundancy of the current preventive maintenance strategy.

For the highway systems, the increase is caused by the introduction of preventive maintenance in 2018 (Figure 2.8). Since 2018, the number of preventive service orders per object per year increased from *confidential* to *confidential* (Figure 2.10a). When a problem is observed during preventive maintenance, a second visit is planned as corrective maintenance. In the case of the highway systems, none of the preventive maintenance activities for highway systems needed a second visit and there have been zero replacement parts installed during the first visits. The number of corrective service orders per object per year remained roughly unchanged, which implies a minimal effectiveness or even a redundancy of the current preventive maintenance strategy. Insight in what type of corrective maintenance activities where planned and whether the introduction of preventive maintenance had any influence on the fluctuating numbers would increase the knowledge on how effective preventive maintenance is for highway systems and whether other types of preventive maintenance could be more effective.

Second of all, there has been an average of *confidential* of orders that could not be finished at first attempt (Figure 2.6). From 2013 to 2014, the percentage of service orders that needed at least three visits has decreased from *confidential* to *confidential*. This sudden decrease

is likely due to a change in registration of preventive maintenance orders. When we exclude 2011 till 2013, the average second visit rate is *confidential*. Since 2017 the second visit rate has increased from *confidential* to *confidential*. A part of the increase is caused by corrective service orders that require at least three visits. Even though the second visit rate is unlikely to become zero, insight in why the orders could not be finished (e.g. spare parts not available or problem description incorrect) would create the opportunity to adapt the corrective maintenance strategy and prevent the second visit rate to increase more.

Third of all, the number of replacement parts needed per object per year has a increasing trend (Figure 2.11a). Insight in why those replacement parts were needed would increase the knowledge on the failure behaviour of the replacement parts.

Based on these insights, we can conclude that the improvement potential that is present requires a better insight in why problems have occurred and how those problems are solved. This information is available in the form of unstructured text fields. To extract that information from those fields, we must conduct a literature review on information retrieval from unstructured data, also known as text mining. Valuable information to retrieve is (1) which problems are present and how much do they effect the performance, (2) how are these type of problems solved, and (3) what characteristics does every problem type have.

2.5. CONCLUSIONS

In this chapter, we learned about the company and her strategy for maintenance on traffic control systems. We also analysed the available data and discussed the potential of these data. We end this chapter with a number of conclusions. These conclusions form the input for the literature review in chapter 3.

The problem cluster we have developed (Figure 1.1) shows a need for more knowledge on why the traffic control systems fail and how the failures are resolved. Also, there is a lack of knowledge on the information value within the available maintenance logs. The performance analysis showed an improvement potential on two aspects that comply with the problem cluster.

The first aspect is the execution of preventive maintenance which is observed to be ineffective. The ineffectiveness comes from the lack of decrease of corrective maintenance actions per object, that should be a result of the observed increase of preventive maintenance actions. Therefore, we can conclude that the current maintenance strategy is not successful. To assess whether this conclusion holds and if another type of preventive maintenance could be successful, first we must require more knowledge on why the systems fail, if the failures could be prevented and how failures could be prevented. This information can be obtained by assessing the problems that have been described in the maintenance logs for corrective service orders.

The second aspect is the slight increase of maintenance requests since 2017 that cannot be solved in one attempt (i.e. the second visit rate). The cause of the new increase is unknown. To adapt the corrective maintenance strategy such that the second visit rate does not increase even more, first we must require more knowledge on why problems could not be solved at first try. This information is also described in the maintenance logs for corrective service orders.

Both aspects describe the same knowledge problem: ”What information is available within maintenance logs for corrective service orders and how can this information be used for new service orders?” A solution to this knowledge problem is the introduction of a classification model. This is a model built with historic data (i.e. the maintenance logs), which can be used to classify new data based on a classification structure. As a structure we will look at how systems fail and create distinctive groups based on the failure types. Each group has its own

characteristics, which provide information on the most suitable problem solving approach. Given a classification model, new data can be classified. This classification can either be used as a knowledge input for a Field Engineer or as a first step in solving the problem. In either way, the goal of such a model is to execute maintenance with a more focused approach. This should lead to fewer redundant visits to an object and shorter visit times, which both lead to lower maintenance costs.

To create a classification model, first we will consult literature in chapter 3 to increase our knowledge on information retrieval from unstructured text fields. Next, we will discuss techniques to create a classification model with such unstructured data. Finally, we will combine the knowledge from literature to propose a model in chapter 4.

3. Literature Review

In chapter 2, we identified the need for a literature review on methods for extraction of information from unstructured data and transformation of this information to a model. The model should classify the transformed data. Each class has its own combination of quantitative parameters like resolution times or number of failures per object per year. The purpose of such a model is to determine the extend to which information retrieval from unstructured data can help the Service Desk at Dynniq during the decision-making process on how to solve a maintenance request. In this chapter we will expand our knowledge by answering the second research question: *"What methods does literature provide for maintenance strategies based on data about maintenance requests and solutions?"*

Extracting information from data and transforming these data into a model is generally known as 'data mining'. We will start this chapter in section 3.1 with the introduction to this subject. Next, in section 3.2 we will focus on a specific field of data mining called text mining. Text mining is about the extraction of information from unstructured text fields. Then, in section 3.3 we will discuss how these structured data can be used to create a classification model. We will end the chapter with conclusions in section 3.4.

3.1. Data mining

Data mining can be defined as the entire process of applying computer-based methodologies for discovery of knowledge from data (Kantardzic, 2011; Larose, 2005; Shearer, 2000). The process includes data cleaning, reduction and preparation steps as well as statistical methods and algorithms for each type of data mining. Due to the comprehensive concept, data mining is often used without a complete knowledge of what the outcome of the exercise will be and what steps need to be taken. Kantardzic (2011) defines data mining as an iterative process in which progress is defined by discovery, through either automatic or manual methods. In this literature review, we focus on the methods for (semi-) automatic knowledge discovery from unstructured data.

3.1.1. Data mining roots

Data mining has its roots in multiple disciplines and with the search for new methods in the core of the data mining philosophy, it has transformed with every new theory or application that is added to its domain (Kantardzic, 2011). Many frameworks for the application of data mining have been developed over the years, such as CRISP-DM, the methodology that we have used to structure this research. Although every method has its own focus, Rogalewicz and Sika (2016) have defined three main steps which are at the core of any data mining project: (1) *pre-processing*, the preparation of raw data such that the data is ready for calculations, visualizations, and transformations, (2) *main-processing*, the build of a model based on calculation and transformations, and (3) *post-processing*, the evaluation and deployment of the model.

3.1.2. Data mining concepts

Data mining concepts like classification and clustering are often incorrectly used or mixed up (Kantardzic, 2011; Shearer, 2000). To address the concepts correctly, we use the concepts that Shearer (2000) has defined as data mining problems. These concepts describe the level of detail which is requested from the data mining exercise. Each concept comes with a set of methods to approach the problem. For this research we follow the five concepts as defined by (Shearer, 2000).

Concept 1 - Data description and summarization
At the start of a data mining process, the precise outcome is often unknown. Also, the data is often incomplete, unclear, or its origin is unknown. Therefore, the first step in understanding the data is to describe the data and summarize what the value of the data might be. Having a basic understanding of the how the data can be transformed narrows the scope of the process. Description and summarization often comes with simple descriptive statistical and visualization techniques. Often, when data have never been used for optimization of a company's processes, this first concept creates enough insight for a board to align their business goals and create a profitable strategy.

Concept 2 - Segmentation
Segmentation is a concept close to data summarization. Segmentation aims at the separation of meaningful subgroups within the data. These subgroups can be related more easily to business questions than a full data set. Segmentation is often a first step towards classification, but can be used as a single concept, for example when only a part of the data needs to be visualized. In this research, we have used segmentation to group the data for the company performance analysis in section 2.3.

Concept 3 - Classification
Classification differs from segmentation in the sense that it assumes that there is a set of objects characterized by some attributes that belong to different classes. For example, the attribute *number of preventive maintenance orders* can be a characteristic for both *electric failures* and *mechanical failures*. With classification, the objective is to build a classification model. The model can use either manual, semi-automatic or automatic methods. A new data entry will be assigned to a *class* via a set of rules. Examples of classification techniques are: *decision trees, rule-based classifiers, support vector machine classifiers, neural network classifiers* and *Bayesian classifiers* (Aggarwal and Zhai, 2012).

Concept 4 - Prediction
Prediction is very similar to classification. The only difference is that the attributes used are usually continuous attributes and always have a structured form. Prediction models are therefore not applicable to text data, which is unstructured. Regression and forecasting are examples of methods that can be used for prediction (Kantardzic, 2011).

Concept 5 - Dependency Analysis
A dependency analysis, which is also only applicable to continuous attributes, is defined by models that describe significant dependencies between data items or events. Such an analysis is based on the combination of predictions from different items and can be used to predict the value of an item, given information of other items. Dependency analyses are seldom significant in large data sets due to the many influences that overlay each other. Therefore, the method is often combined with segmentation to focus the analysis on a smaller data set.

3.1.3. Conclusions

In this research, we will focus on the concept of *classification*. Shearer (2000) recommends to take one step at a time in improving data mining models within a company. Dynniq is currently not using any models for data mining, but the data is collected such that a proper description of the data can be made and the data can be easily segmented. Therefore, the next step is a classification model. Prediction and dependency analysis are concepts for future projects which involve the use of continuous attributes from the dataset.

In the next section we will look at the data mining field called text mining. Text mining answers the question on how to extract information from unstructured text data. After the introduction of text mining, we will dive into classification methods that use text data as an input for the model.

3.2. TEXT MINING

Text mining is an expertise within the field of data mining (Kantardzic, 2011; Larose, 2005). Kantardzic (2011) presents text mining as a conceptual framework with two phases: text refining and knowledge distillation (Figure 3.1). The first step, text refining, transforms unstructured text in a chosen intermediate form (IF). This IF can for example be the extract of keywords from the original text. The IF is a cleaned, structured form of the original text field. The second step, knowledge distillation, deduces patterns or knowledge from an IF. Text mining on its own results in a cleaned, structured form of the text data, which can be used in two ways. The first way is to use data for document organization, visualization, and navigation. The second way is to use data for text-analysis functions like summarization, clustering or classification. In this research, we will focus on the classification of unstructured data.

Even though knowledge can be distilled from data fields, text mining only looks at the presence of words. When a word or group of words is present in a text, the text gets a certain value. The more a word or group of words is present in a text, the stronger the influence of that word or group of words is on the value of a text. A field close to text mining is called Natural Language Processing (NLP). NLP comprehends the ability to combine complex linguistical aspects, like recognizing semantic patterns, relationships and human emotion in both text and speech. So, where NLP looks at meaning and relationships in text, text mining looks at the presence of words.

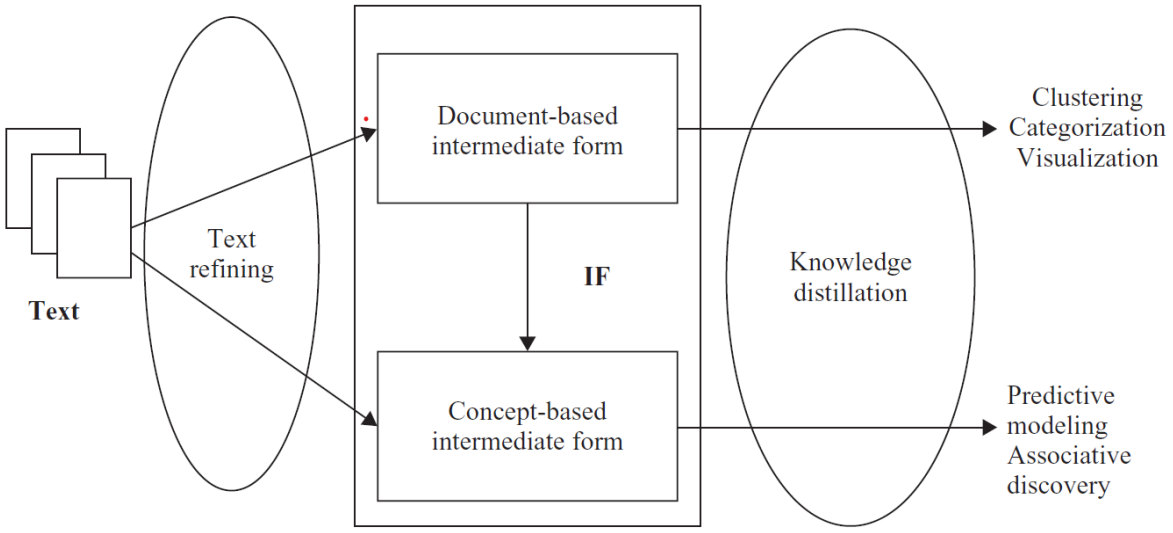


Figure 3.1: Text mining as a framework

Kantardzic (2011) states that studies from 2011 have indicated that 80% of a company's information is contained in text documents. And as the most natural form of storing information is text, text mining is believed to have a commercial potential even higher than that of traditional data mining with structured data. Recent practical applications of text mining are known from a wide range of fields such as health care (Sen et al., 2019), maintenance (Edwards et al., 2008; Wei et al., 2017), online purchasing (Reshmy and Paulraj, 2015), and smart city innovations (Tan et al., 2015).

In the remainder of this section, we will focus on what Kantardzic (2011) refers to as the two most commonly used models for text mining applications, the Bag of Words (BoW) model and the Vector Space Model (VSM).

3.2.1. Bag of Words model

The first computerized model for text mining was the 'Bag of Words' (BoW) model developed by Salton and McGill (1983). The model is assumed to be developed in the early 70's, but has not been published until 1983. The BoW model is developed to compare texts such as documents or sentences. The model counts the number of times each word is present in the text and adds these numbers to a 'bag'. Beforehand, words are often transformed to the stem of the word (e.g. "failed" becomes "fail", "switched" becomes "switch"), which is the basic form used as a dictionary entry. When a certain BoW is present in two separate documents, it is assumed that these documents are of a similar context.

As an example we can use a simple sentence: "System failed, system is switched off". We will use this example more often throughout the report. The example is in English, even though the data set is in Dutch. We use the English example to improve readability. This sentence is transformed to a list of words in the stemmed form: ("system", "fail", "system", "is", "switch", "off"). Then, a BoW is created where the words are counted:

$$BoW_1 : \{\textbf{"system"} : 2, \textbf{"fail"} : 1, \textbf{"is"} : 1, \textbf{"switch"} : 1, \textbf{"off"} : 1\}$$

In a BoW model, the order of words is non-specific, but often the BoWs are being linked to a full list of words (i.e. a vocabulary) with a specific order. This creates the option to shorten the lists into a numerical form. Considering only the aforementioned words, the BoW becomes:

$$BoW_1 : \{2, 1, 1, 1, 1\}$$

The numerical form of the BoW model can be used to find information such as the number of times a certain combination of words is used in a multiplicity of documents or sentences.

3.2.2. Vector Space Model

The BoW model was quickly extended by Salton et al. (1975) into the Vector Space Model (VSM). VSM is based on the BoW model, but considers the use of a continuous scale instead of the discrete form of a BoW. This provides the option of partial matching (e.g. a document has a 60% correlation with a predetermined group) and ranking documents on relevance to a target value (e.g. ranking on correlation). VSM considers documents as vectors in a t-dimensional space. Each document *d* consists of a group of terms *t*. Each term can be a single word or a group of consecutive words. Terms are created after the text is cleaned from grammatical mistakes and stop words. Then, usually an occurrence filter is used to ensure the model is created only with popular terms, which are likely to be used again in new data.

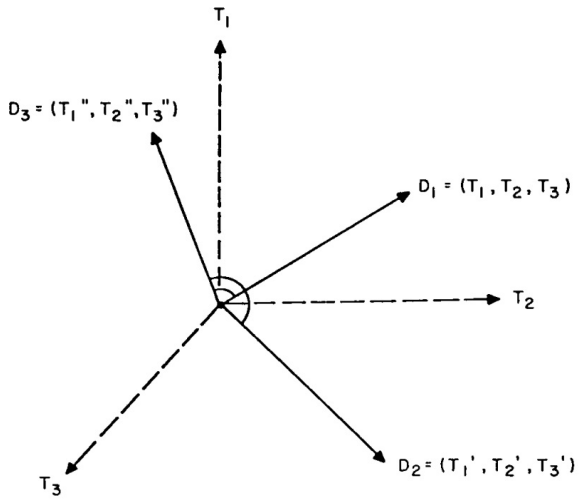


Figure 3.2: Three-dimensional vector space representation

With Figure 3.2, Salton et al. (1975) proposed a visualisation of a 3-dimensional space, which considers document vectors (*D1, D2, D3*) of three different terms (*T1, T2, T3)*. A vector in this space is defined by the terms and their term weight. The general vector equation is defined as follows:

$$\mathbf{v}_d = [t_{1,d}, t_{2,d}, ..., t_{T,d}] \tag{3.1}$$

where $t_{t,d}$ is the term weight of term *t* in document *d*. The term weight can be: (1) a binary value indicating only the presence of a term, (2) a term frequency value indicating the number of times a term is present in the assessed document, or (3) a Term Frequency $\times$ Inverse Document Frequency value (TF$\times$IDF), which is a term weight that considers all documents assessed as a global parameter and is according to Beel et al. (2016) used in 83 % of text-based recommender systems. The TF$\times$IDF value is determined as follows:

$$t_{t,d} = TF_{t,d} \times log \frac{|D|}{|d' \in D | t \in d'|} \tag{3.2}$$

where $TF_{t,d}$ is the term frequency of term *t* in document *d* and $log \frac{|D|}{|d' \in D | t \in d'|}$ is the inverse document frequency with $|D|$ being the total number of documents and $|d' \in D | t \in d'|$ being the number of documents containing term *t*. As the number of document containing term *t* can never be higher than the total number of documents, the logarithm will be taken over a value larger or equal to one and therefore can never take a negative value. A high term weight correlates to a term with a high frequency and a low number of documents that hold the term. In other words, the term is present often in a few documents, but never present in the other documents. This creates a highly unique feature of the documents that hold the term.

For example, the first word from the aforementioned example ”System failed, system is switched off” is ”System”. If this word occurs twice in a document, the Term Frequency is two. If that word occurs in no other document and we have a total of five documents, the Inverse Document Frequency becomes $log(\frac{5}{1}) = 0.699$. The TF$\times$IDF value becomes $2 \times 0.699 = 1.398$. For the Term Frequency we can also use the relative frequency instead of the absolute value. Then we compare the occurrence of a word and divide it by the total number of words in the document.

With this VSM method, document are considered to be similar when the document vectors are close together in the vector space. Ideally, clear groups are formed within the vector space, which indicates a clear division between document types.

3.2.3. CONCLUSIONS

Most algorithms used for information retrieval from unstructured text data are either based on the BoW or the VSM approach. Both methods transform textual data into embedded solutions. The methods require data that have already been cleaned from linguistical errors like grammatical mistakes. The outputs of the BoW and VSM approaches are a structured representation of text and can be used as input for a classification model. Often, the BoW and VSM approaches are built into the classification algorithm. In the next section, we will discuss such classification algorithms that we will be using in this research.

3.3. CLASSIFICATION MODEL

As discussed in section 3.1, the concept that is most suitable to the available dataset is classification (Shearer, 2000). Aggarwal and Zhai (2012) define the design of a classification model as a five-step plan: (1) creation of classes, (2) assigning classes to a training (data) set, (3) designing a model, (4) training the model with the training set, and (5) validating the model with a validation set. In this research, we will use a supervised learning method to create the classes of the classification model. In a supervised learning method, the classes are based on a theoretical structure rather than created by the model itself. In this research we will base the structure on types of failure, also known as failure modes. The created classes then must be linked to a training set. This set will be used by the classification algorithm to learn the value of a class. This step is essential because, as mentioned in section 3.2, words have no meaning in text mining.

3.3.1. FAILURE MODES

First, we will discuss the theory on failure modes. We will use this theory to create the classes for the classification model, which we will discuss further in chapter 4. As we concluded in chapter 2, the available unstructured data is a description of why a system has failed and how the failure is resolved. In general, a failure occurs when a load on a system is higher than its load-carrying capacity (Figure 3.3). Tinga (2013) defines a failure as reaching such a state that the intended function of the part or system can no longer be fulfilled. This does not always mean a physical failure, like fracture or melting. Tinga (2013) also defines a clear distinction between a failure mode and a failure mechanism. A failure mode is the manner in which a system or component functionally fails, a failure mechanism is the physical or chemical process or mechanism yielding degradation of the component and ultimately leading to the physical failure.

Next to physical failures, problems are often caused by usage (Figure 3.3, secondary load). Human errors or wrong procedures can increase the total load to a higher state than the load-carrying capacity. Failure modes are often used in reliability related analyses like an Failure Mode Effects and (Criticality) Analysis (FME(C)A) or a Root Cause Analysis (RCA).

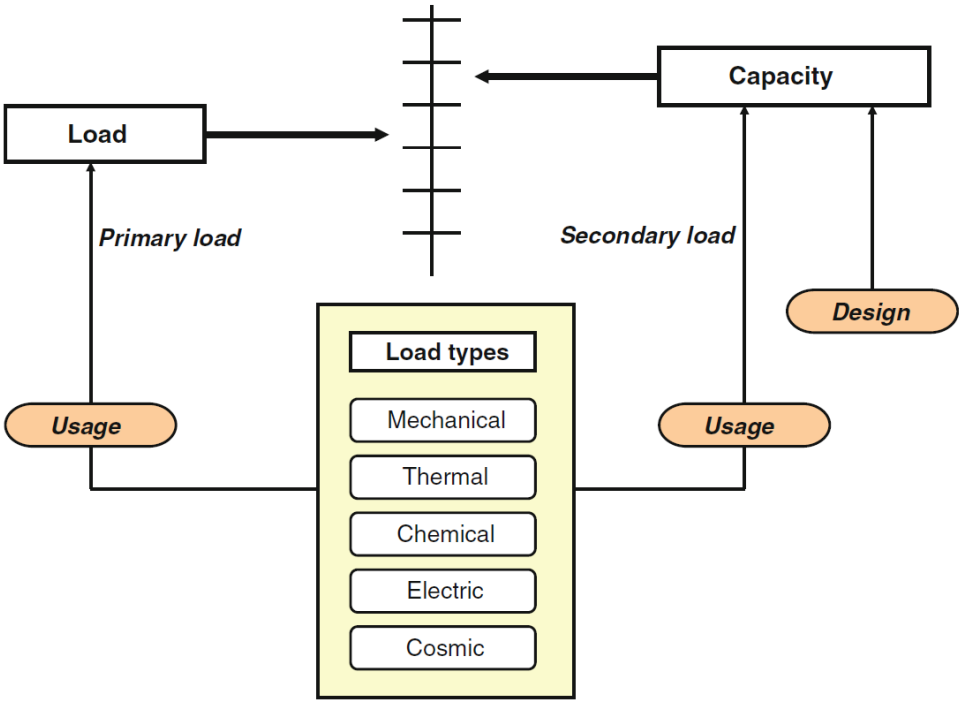


Figure 3.3: Schematic representation between load and load-carrying capacity

Failure mechanisms as such are not described in the available maintenance log, because a failure mechanism often cannot be observed with a simple investigation. Therefore, we will look only at the failure modes. Failure modes can have different levels of hierarchy, often based on the system's breakdown structure. For example, a car can have the failure mode 'non-working engine', the engine then can have a failure mode 'broken crankshaft' and the crankshaft can have the failure mode 'worn-out linkage'. For this research, we will focus on the failure modes that we can deduce from the text data using the BoW approach. As mentioned in chapter 2, new data will be assigned to a class (i.e. a failure mode) and this class has its own characteristics. In this way the classification model will help a Field Engineer to define a more focussed problem solving approach.

3.3.2. CLASSIFICATION TECHNIQUES

To design the classification model, we will compare multiple classification methods. The most commonly known types of classification methods, as described by Aggarwal and Zhai (2012), are decision trees, rule-based classifiers, support vector machines, neural networks, and Bayesian classifiers. Decision trees and rule-based classifiers are both based on decision rules and work best in a hierarchical classification structure (Edwards et al., 2008). These classifiers are in general hard versions of classifiers. As Aggarwal and Zhai (2012) explain, in the hard version of the classification problem, a particular label (e.g. electrical failure) is explicitly assigned to the instance, whereas in the soft version of the classification problem, a probability value is assigned to the instance (e.g. 60% electrical failure, 40% software failure). Hard classifiers can become soft when multiple data entries are used. This is the case for text data with at least two words. For example, the earlier used sentence "System failed, system is switched off" could have one word that describes an electrical failure (e.g. "failed") and two others that describe a software failure (e.g. "switched off"). In this case, the sentence has a partial match with each of the two classes.

Support vector machines (SVMs), neural networks and Bayesian classifiers are all soft classifiers. Also, a hierarchical classification structure is not applicable. SVMs and Bayesian classifiers are easier to implement than neural networks due to the requirement of a smaller training- and validation set. Neural networks require more training, but have also proven to be more accurate for text classification purposes (Aggarwal and Zhai, 2012; Wei et al., 2017).

For this research, we have chosen the classifiers based on the following assumptions: (1) the research is an exploratory research on how to retrieve knowledge from a dataset, (2) algorithms for the classifiers must be available within a free software solution, and (3) the classification structure (i.e. the created structure of classes) is based on a theoretical foundation (section 3.3.1). These assumptions create the need for a conceptual classification model which proofs whether information is available within the dataset and what the value of this information is. With this in mind, two types of classifiers, the decision tree and the SVM are most suitable. We have chosen these classifiers because both have a low threshold for training new models and the classifiers are build on different methodologies. Decision trees are based on the BoW model and classify documents one by one according to a consecutive set of rules. SVM classifiers are based on the VSM and look for a significant difference between terms in a large group of documents. We add a third classifier, the neural network, because of its high performance potential. In practice, most classification algorithms are compared on three parameters: the speed and accuracy of the algorithm and whether the algorithm is applicable to small data sets (Aggarwal and Zhai, 2012; Kantardzic, 2011). We collected scores for the defined parameters in Table 3.1 given theory (Kantardzic, 2011) and practical applications (Aggarwal and Zhai, 2012; Edwards et al., 2008; Sen et al., 2019; Wei et al., 2017).

In the remainder of this chapter we will address the algorithms that we will use in our model design in chapter 4.

Table 3.1: Overview of available classifiers for text classification

Document type	Speed of the algorithm	Small training set applicable	Accuracy of the algorithm
Decision Tree	++	++	+
Rule-based	++	++	+/-
SVM	+	++	+
Bayesian	+	++	+/-
Neural Network	- -	+/-	++

Decision tree classifier - C4.5

The decision tree classifier is known as the most widely used logical method for producing classifiers from data (Kantardzic, 2011; Aggarwal and Zhai, 2012). A decision tree classifier uses a hierarchical model for supervised learning to find a local optimum through decision nodes with test functions, also known as test nodes (Figure 3.4). Aggarwal and Zhai (2012) state that most decision tree algorithms are variations of the ID3, C4.5 and C5 algorithms. For this research, we will use the C4.5 algorithm, which is the most advanced version that is available in free software.

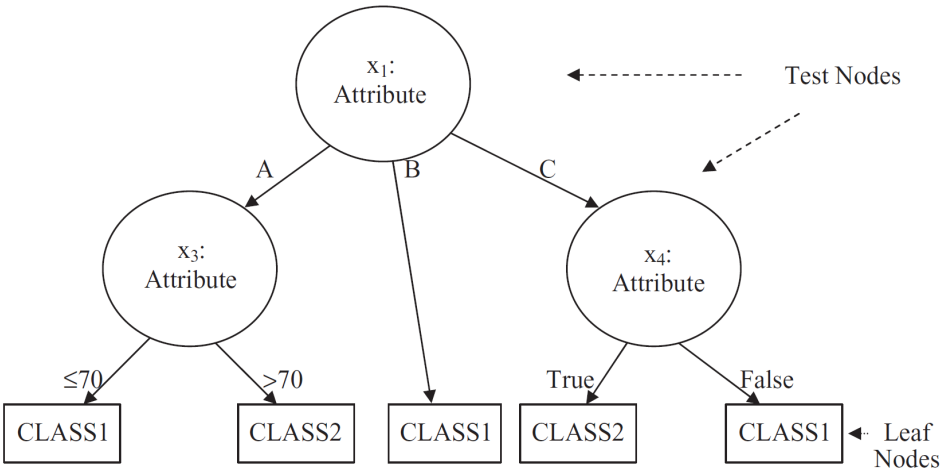


Figure 3.4: Decision tree framework

The C4.5 algorithm is developed by Quinlan (1993), who also developed the ID3 algorithm (Quinlan and Rivest, 1989), the predecessor of C4.5. The main extension of the new algorithm is the use of the stemmed form of words (e.g. ”failed” becomes ”fail”, ”switched” becomes ”switch”) instead of all word variations. C4.5 uses the concept of normalized information gain to decide which downstream branch is most suitable to the information, also known as Gain ratio. The gain is determined by finding the difference in information entropy between the current node and the possible nodes downstream. Information entropy can be defined as the degree of disorder in a spatial area. The goal is to find a downstream node with the lowest possible entropy. Nodes have a high entropy if no or a partial match (i.e. a high disorder) is present. For instance, when a document that consists of ten terms has a match on three of the terms in node *A* and five of the terms in node *B*, the match with node *B* is better and therefore the entropy is lower. The information gain towards node *B* will be higher as we are looking for the lowest possible entropy value. An alternative to the information gain is the Gini index. The Gini index looks for the probability of a sample being misclassified if we randomly pick a label regarding the distribution in a branch. The two approaches are in fact very similar and according to Raileanu and Stoffel (2004), they only disagree 2% of the time.

The root or top of the tree is simply the fact that a failure has occurred. Downstream is a set of nodes that divide all possible words into groups that belong to a certain high level failure mode, for example 'electrical failure' or 'mechanical failure'. Even more downstream the group 'electrical failure' could be split into groups like 'failed wire' or 'failed module'. Splitting all available terms is done by a decision tree learner. The learner uses a training set of which the classes are known. The learner will then create a decision tree such that each instance in the training set corresponds with a class. The first node in a subtree that correlates well enough with a certain class is called a leaf node (see Figure 3.4). This node is the final, most downstream node of a subtree. The leaf nodes are found by a technique called pruning, which can be defined as reducing the size of decision trees by removing sections of the tree that provide little power to the classification of instances (Aggarwal and Zhai, 2012). Pruning is used to prevent over- or under fitting, which both lead to a difference in class distribution between data sets. Pruning can be done by holding out a part of the data as a separate training set. When the class distribution of the training set for pruning does not correspond with the class distribution of the training set for decision tree building, the leaf node is pruned.

When the steps are executed, the decision tree model must be validated by a validation set. The validation set is run through the model and each instance is classified. The outcome is compared to a manual classification of the validation set and a score is determined. The score is usually the percentage of instances that are assigned to the wrong classes (i.e. misclassification) or the right classes (i.e. accuracy).

SVM Classifier - Sequential Minimal Optimization

The main principle of SVM classifiers is to determine separators in a multidimensional search space, which can best separate the different classes (Aggarwal and Zhai, 2012). Each document forms a data point in the search space. The location of the data point is given as a vector (recall section 3.2.2). The separators, which are also vectors, form boundaries between data points which leads to the formation of groups of data points, for example the circles and crosses in Figure 3.5. Each group corresponds with a class label. Separators are also known as decision boundaries. An optimal separator has the largest normal distance to any of the data points in the search space. The advantage of a large normal distance, which can be seen as a margin around the separator, is that the separator is still valid when small changes in the data set occur. This increases the feasibility of new data sets into the model. As an example, in Figure 3.5, three different separators are shown. Separator *A* has the largest normal distance to any data point and is therefore the most optimal separator. Separator *A* can be optimized by shifting *(B)* and rotating *(C)*.

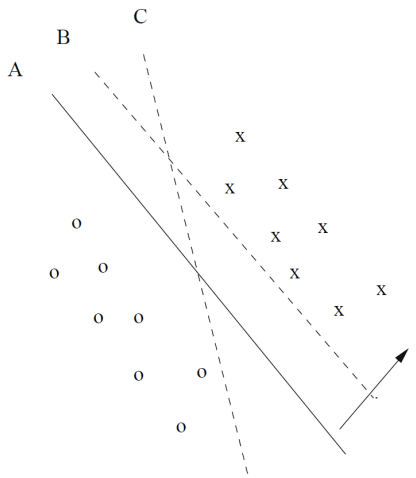


Figure 3.5: SVM visualization with separators A, B, and C

Finding the optimal separator is an optimization problem called a Quadratic Programming (QP) problem. A commonly used SVM algorithm that solves the QP problem is the Sequential Minimal Optimization (SMO) algorithm developed by Platt (1999). SMO simplifies the QP problem to a multiple of two-dimensional problems, such as Figure 3.6. The goal of SMO is to find the vector that describes the optimal separator. To do so, first we must find the vector that describes the minimal distance between two data sets of different classes. The optimal separator is the perpendicular to that vector. A perpendicular of two vectors can be described as follows:

$$\vec{w} \bullet \vec{s} = \|\vec{w}\| \ \|\vec{s}\| \cos 90 \deg \tag{3.3}$$

with $\vec{s}$ being the separator and $\vec{w}$ being the vector that describes the minimal distance between two data sets of different classes. As $\cos 90 \deg$ is equal to zero, the equation becomes:

$$\vec{w} \bullet \vec{s} = 0 \tag{3.4}$$

The vector that describes the minimal distance between two data sets of different classes can be found by solving the following minimization problem:

$$\min_{\vec{w}} \|\vec{w}\|^2 = \min_{\alpha} \|h^+ - h^-\|^2 = \min_{\alpha} \| \sum_{d \in +} \alpha_d \vec{d} - \sum_{d \in -} \alpha_d \vec{d}\|^2 \tag{3.5}$$

subject to:

$$\alpha_d \geq 0 \tag{3.6}$$

$$\sum_{d \in +} \alpha_d = \sum_{d \in -} \alpha_d = 1 \tag{3.7}$$

where $\vec{w}$ is the weight vector that describes the minimal distance between two data sets of different classes. This can also be described as the distance between data points h^+ and h^-. These points are located on the edge of the positive and negative polytopes, respectively. A polytope is the area in which all data points of a single class are located. A polytope is always convex, meaning that any two data point in the polytope can be connected with a straight line without leaving the area. In Figure 3.6, the green area is the positive polytope and the blue area the negative polytope. Each point in the polytope, for example x_1 in the green area consists of a combination of the corner points of that polytope. So is x_1 equal to $\frac{1}{2}d_1 + \frac{1}{2}d_2$. Furthermore, α_d are the Lagrange multipliers, which are used to find the closest minimal distance and the sum of all Lagrange multipliers are always equal to one. In the case of x_1, we have two Lagrange multiplier, which are both equal to $\frac{1}{2}$. Finally, vector $\vec{d}$ is the document vector that represents a data entry with term weights for each term in its vector.

The SMO algorithm starts its calculation in the centroid of the polytope. That point has a relative distance to each corner point, for example 1/3 to each of three corners. This relative distance is the Lagrange multiplier α_d and the corner point is the document vector $\vec{d}$. Two of the Lagrange multipliers are chosen at random. The multipliers are changed, holding equality constraint 3.7 intact. The best possible combination is chosen for which the weight vector $\vec{w}$ is minimized the most. These values are set and a new pair of multipliers is chosen. When an optimal separator is found, the algorithm stops. Because of the large number of combinations for the two multipliers, simple heuristics are often used to find an optimum between α_1 and α_2 and decrease the calculation time. Keerthi et al. (2001) state that SMO is often much faster and

has better scaling options than other SVM algorithms. SMO is also much easier to implement due to the choice of either linear or non-linear separators. Keerthi et al. (2001) have improved the SMO algorithm by increasing the algorithm's efficiency even more.

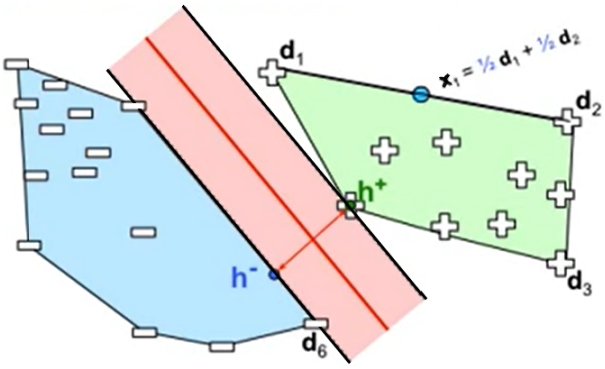


Figure 3.6: SVM visualization with positive polytope (green), negative polytope (blue) and margin (red)

In some data sets, the data points have so much overlap that the data set is evidently inseparable. When this occurs, non-linear separators are required. The calculation of these separators can be simplified by the use of a kernel function. This function is a method of using linear classifiers to solve a non-linear problem. In other words, the kernel function is a shortcut to do certain calculations faster which otherwise would involve computations in a higher dimensional space. Many kernel types are available, each with their own applicability to mathematical problems.

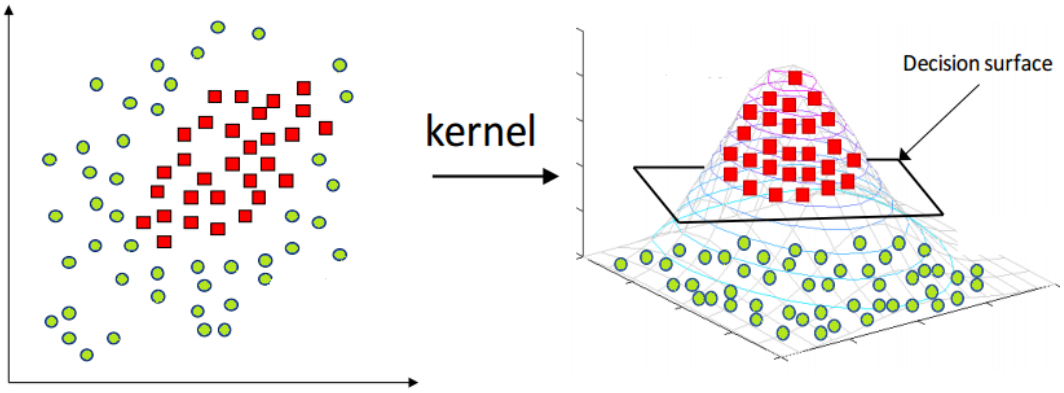


Figure 3.7: Transformation of a two-dimensional space into a three-dimensional space by use of a kernel

To illustrate the function of a kernel, we can use Figure 3.7. The Figure shows a simple way to separate the red squares from the green circles. We just add a third dimension to the dataset. It is also possible that a fourth, fifth or higher dimension is required to create such a linear separator. This transformation can be expressed by the following equation:

$$K(x, y) = \phi(x) \bullet \phi(y) = \phi(x)^T \phi(y) \tag{3.8}$$

where $K(x, y)$ is the kernel and x and y are n-dimensional inputs (i.e. the document vectors). ϕ is a $n \times m$ dimensional space, with m being the required dimension for separation (e.g. the third

dimension in Figure 3.7). Normally a function like $\phi(x)^T\phi(y)$ requires us to first calculate $\phi(x)$ and $\phi(y)$ and then calculate the dot product of the functions. This can be simplified by using a corresponding kernel. As an example, we can take the following two-dimensional vectors:

$$x = (x_1, x_2)^T \tag{3.9}$$
$$y = (y_1, y_2)^T \tag{3.10}$$

Lets say a linear separator is possible in the fourth dimension (i.e. $m = 4$), then both $\phi(x)$ and $\phi(y)$ get four dimensions. Equation 3.8 becomes:

$$K(x, y) = (x_1^2, x_1x_2, x_2x_1, x_2^2)^T(y_1^2, y_1y_2, y_2y_1, y_2^2) \tag{3.11}$$

This equation can be simplified to a two-dimensional form, the kernel, which reaches the same results:

$$K(x, y) = (x^T y)^2 \tag{3.12}$$

The existence of this simplified form is called the kernel trick. See appendix A for a numerical example of the calculation. In this research we will use the available kernels in the software and evaluate the performance of each of these kernels. Performance can be low if a kernel function is not able to separate all data points correctly.

Neural network - Artificial Neural Network

The third algorithm is a neural network. Neural networks come in many shapes and forms (van Veen, 2017). For this research we will focus on a simple Artificial Neural Network (ANN). ANNs are based on the function of the human brain. In the human brain, billions of neurons are interconnected and form a network. Each neuron holds some information, which is influenced by all neurons that are connected to this first neuron. A neural network is similar, but often presented in a form that computers understand. Figure 3.8 shows an input and output layer, which are the information that is provided by the user and result from the model, respectively. In between these layers is a hidden layer. Here, the information from the input is translated by some mathematical functions to a proper output. This form of neural networks are called feed forward models. This means that information can only go from the input towards the output (i.e. from left to right).

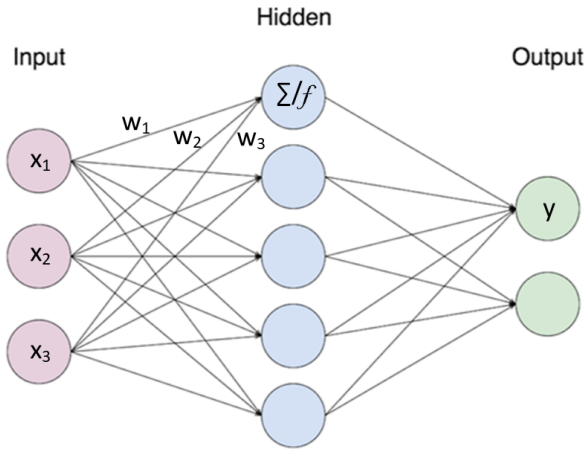


Figure 3.8: Visualisation of an ANN with one hidden layer

The transformation of the input values within the hidden layer follows three steps (Figure 3.9). In the first step, the input values (x_1, x_2, x_3) are multiplied by the assigned weights (w_1, w_2, w_3). The weights are the values that are altered during the optimization of the model. At the start of building the model, the weights are chosen at random. This is only done in the first iteration. In the second step, the multiplications are added $(x_1w_1 + x_2w_2 + ...)$. At this point, the summation can be altered by a predetermined bias, which is added to the summation. This is a user determined value that influences the performance of the model. The bias is often used for optimization, although the model can work without a bias. It can only be chosen ones, before the model is built. After summing up the input values, the summation is often changed to a value between 0 and 1. This third step is called the activation function. The most common activation function is the sigmoid function, which has the following equation:

$$sig(t) = \frac{1}{1 + e^{-t}} \tag{3.13}$$

where *t* is the sum of all weights supporting a neuron in the hidden layer.

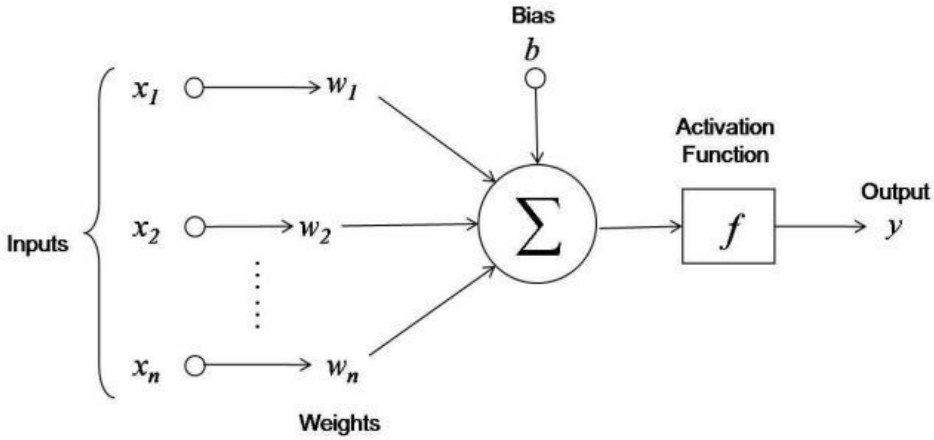


Figure 3.9: Visualisation of the calculations in the hidden layer of an ANN

With these three steps, the first iteration of the neural network is finished and we have an output value. This output value is usually not equal to the actual value that is predetermined and therefore it must be optimized. The optimization method that is used most often is called back-propagation. This method is originally developed by Rumelhart and McClelland (1986). For our research, we will take the back-propagation algorithm as a black box as we cannot adjust the parameters of this function. A full description of the back-propagation algorithm used in this research is developed by Riedmiller and Braun (1993) and is called RPROP, which stand for 'Resilient Propagation'. Back-propagation is a method that looks at the difference between the output of the model and the predetermined value. The difference is called the error. When we use this error and inverse the steps from Figure 3.9, we are provided with the error of the hidden layer and the error of the input value. Knowing these errors, we know which connections in the neural network provide the highest error. The weights of these connections are adjusted and the three steps from Figure 3.9 are recalculated. Repeating the back-propagation method multiple times will decrease the error and increase the accuracy of the model.

In the case of our research, we are using text data. To be able to do the calculations mentioned, we use document vectors (recall section 3.2.2) as input values and classes (i.e. failure modes) as output values. What the hidden layer does is to find appropriate weights such that each document vector has a certain influence on the outcome. For example, when a sentence has a

clear description of an electrical failure, the weight from that input sentence towards the output label ‘electrical failure’ will be high.

3.3.3. Conclusions

We have proposed three classification algorithms that are applicable to unstructured data and comply with the assumptions: All algorithms are available in free software solutions and relatively easy to implement in new models. All algorithms approach the classification issue differently, which will be compared in the validation of the classification model (chapter 5).

3.4. Conclusions

In this chapter, we have performed a literature review to answer the second research question: *”What methods does literature provide for maintenance strategies based on data about maintenance requests and solutions?”* We end this chapter with a number of conclusions regarding the research question. These conclusions form the input for the model design in chapter 4.

In the literature review, we discussed methods for the extraction of information from unstructured text fields from maintenance logs and the transformation of that information to a model. We have learned that the extraction of data, called data mining, can be executed in five levels of detail. Reflecting those levels of detail on the available data set and the problem cluster, we can conclude that the concept of classification is most applicable for this research.

The data on which we will classify Service Orders are unstructured text data. The field of data mining that resolves the extraction of information from unstructured data is called text mining. We can conclude that most text mining methods are a variation of the Bag of Words (BoW) or Vector Space Model (VSM) approaches. Given these approaches, we have concluded that the most applicable classification algorithms are the decision tree algorithm ’C4.5’, which is based on the BoW model, the SVM algorithm ’Sequential Minimal Optimization (SMO)’, which is based on the VSM, and the ANN algorithm ’Resilient Propagation’ (RPROP), which is also based on the VSM. We have chosen two out of three algorithms (C4.5 and SMO) because of their ease of implementation in newly introduced models, their availability in free software and the option to use a theoretical foundation for the creation of classes. We added the RPROP algorithm because of its high performance potential. The theoretical foundation that will be used is the theory of failure modes. This theory complies with the available text data and is possible to implement in a hierarchical form.

The literature review has provided the required theory and algorithms to create a classification model. What the literature review cannot provide is a vocabulary that holds all terms that are used within the company. Also, an implementation plan is required on how the model can be used by the company. We will address these queries in the remainder of this research.

4. Model Design

With the theory from the literature review (chapter 3) known, we can discuss the design of our classification model. In this chapter, we will give answer to the third research question: ”How can data be modelled to provide guidelines for optimizing the maintenance strategy?”. We start with a general description of the model in section 4.1 to get a high level understanding of the model. Next, we will discuss the test and validation sets and the class labels that are assigned to these sets in section 4.2. Then, we will discuss the model in more detail, starting in section 4.3 with the cleaning of the data set, the selection of terms that will be used in the model, and transforming the cleaned data into document vectors. Next, in section 4.4 we will explain how the decision tree, SVM, and ANN algorithms are used to build the text classification model. After the build of the model, we will discuss the validation of the model in section 4.5 and the use of the model for new data in section 4.6. We conclude the chapter in section 4.7.

4.1. General model description

Each section of this chapter corresponds with a step in the design of the classification model. The model is designed with the use of Knime Analytics, a data mining tool which is based on Object Oriented Programming (OOP) and is free to use. The model is designed to use unstructured text data, transform these data such that it can be used to create a classification model and verify the model trough a validation set. The output of the model is an accuracy score, which is the percentage of correctly classified instances. The model consists of two workflows. In the first workflow (Figures 4.1 and 4.2), the classification models are built and tested on their accuracy. In the second workflow (Figures 4.3 and 4.4), new data is run through the models created in workflow A to assign a class to each data entry.

The model is built as a proof of concept, which is the first step towards an automated model that uses a live connection with the current ERP system Navision. In chapter 6 we will discuss the required steps towards this fully implemented system.

4.1.1. Workflow A - Building the classification model

The first workflow can be split into four steps (Figure 4.1 and 4.2). In the first step (*loop preparation*), the unstructured data is read from a CSV file and a list of system types (e.g. EC-1 or OS-2) is created. The system type corresponding to each data entry is available in the input file. Then a counter loop is activated, which selects one system type from the aforementioned list per iteration. Only the data entries that comply with the selected system type are then run through the loop. The loop will run through steps two (*data preparation*) and three (*data modelling*) and save the created models for each system type separately. These models can be used later on for new data entries. In the second step (*data preparation*), the data is transformed to a form that is readable for the classification algorithms. Here, we clean the data from grammatical errors and transform them into document vectors (recall section 3.2.2). We also make a selection of terms that are most valuable for the building a classification model. This step is called feature selection and removes for example stop words (i.e. words that are used often but have no value) from the text data. The resulting document vectors are used as input data to create the classification models.

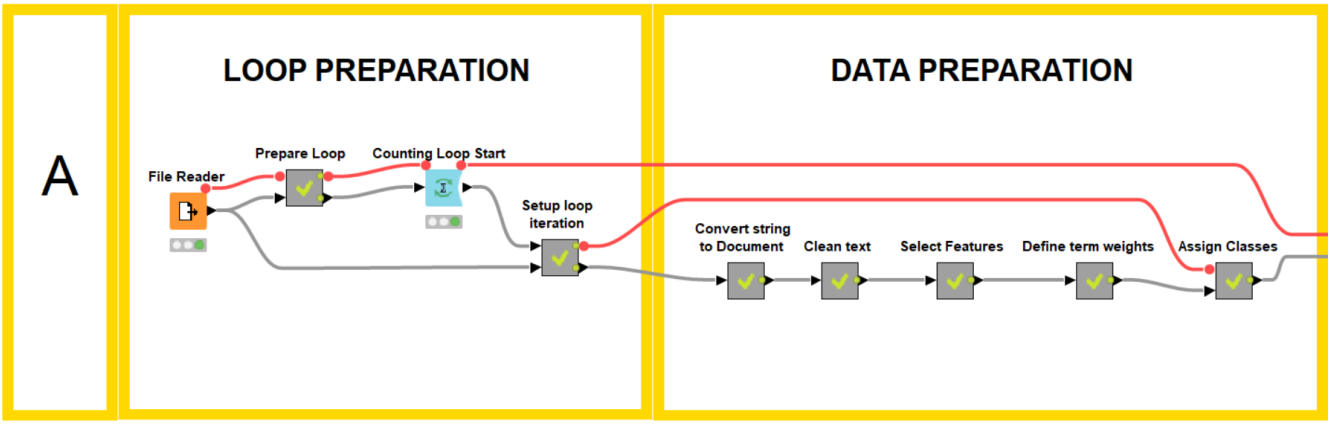


Figure 4.1: Workflow A - part 1: Building the classification model

The models are built and optimized in step three (*data modelling*). Here, we partition the data into a training and validation set, which are used to build and validate the models, respectively. For testing, a loop is added which can rebuild each model with different random numbers, also known as seed numbers. The random number is used to randomly partition the data set into a training and validation set. Performance of the models is tested with the accuracy score. The model and the accuracy score are saved and the counter loop returns to step two (*data preparation*). When the loop is completed, the accuracy results are grouped and visualised in the fourth step of the workflow (*data visualization*). One of the visualisations is a scatter plot with the accuracy scores for all models (three model types *SVM, DT, ANN for* each system type). Another visualization is the decision tree view, which shows the decision tree that is built during each step.

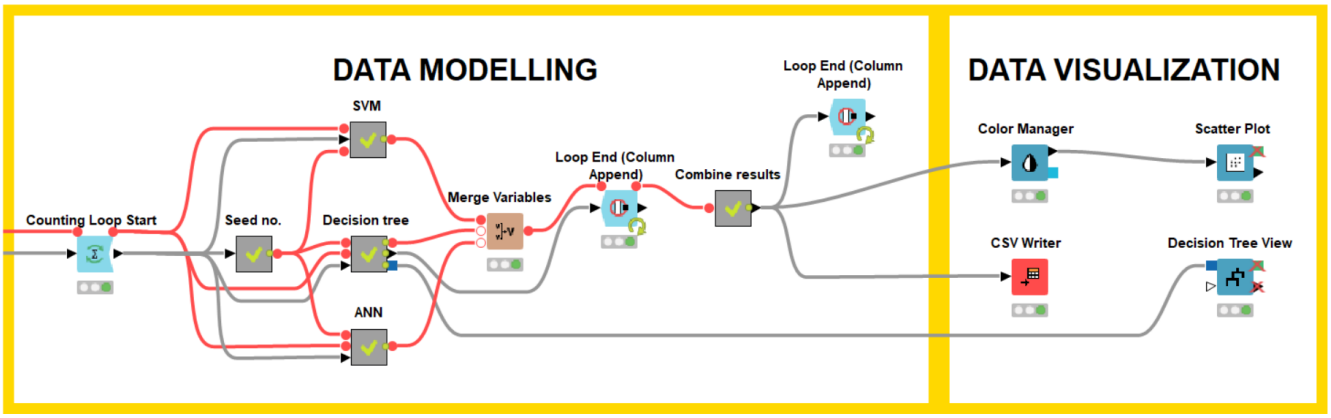


Figure 4.2: Workflow A - part 2: Building the classification model

4.1.2. Workflow B - Classification of new data

The second workflow (Figures 4.3 and 4.4) can be split into four steps as well. The first two steps (*loop preparation* and *data preparation*) are similar to the steps from workflow A. Again a counter loop is started which runs one system type per iteration, but only the system types present in the new data set are considered. So if only system types EC-1 and OS-2 are present in the new data set, only two iterations are executed, one for each system type. Then the data is cleaned and selected the same way as in workflow A.

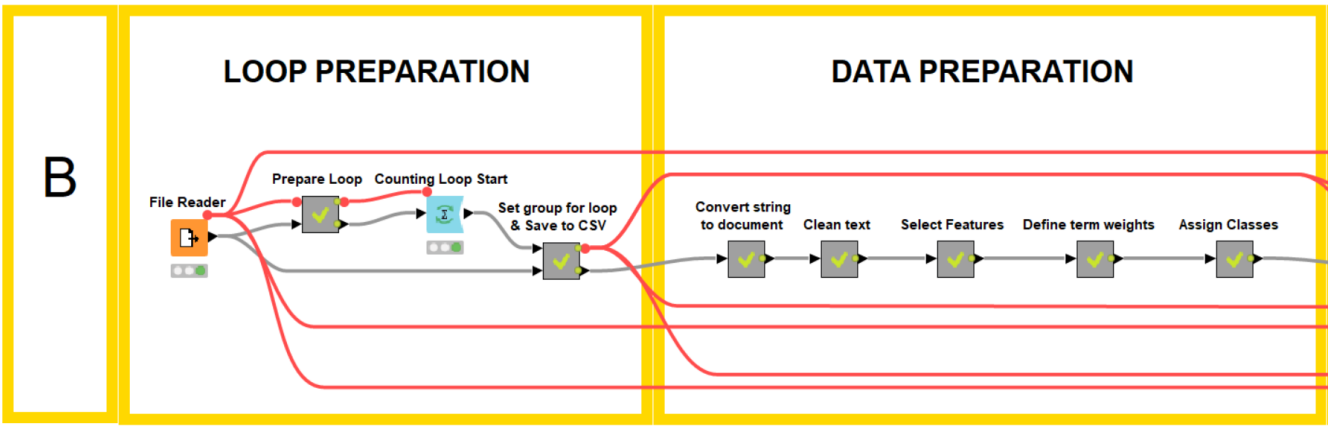


Figure 4.3: Workflow B - part 1: Classification of new data

In step three (*data modelling*), a dictionary is created. This dictionary holds all terms from the training, validation, and new data set and is required for the model to classify instances based on their terms. Recall from section 3.2 that a term can be a word or group of words. In this model, we only consider single words. The current data set is too small to expand to groups of words, because groups of words are even more unique than single words and the chance of terms recurring in new data entries becomes smaller when more unique terms are used. The workflow then chooses the model that corresponds to the system type of the current loop iteration and the new data is classified. Finally, in the fourth step (data visualization), the resulting data set is saved to an excel file for further use in the dashboard (section 4.1.3)

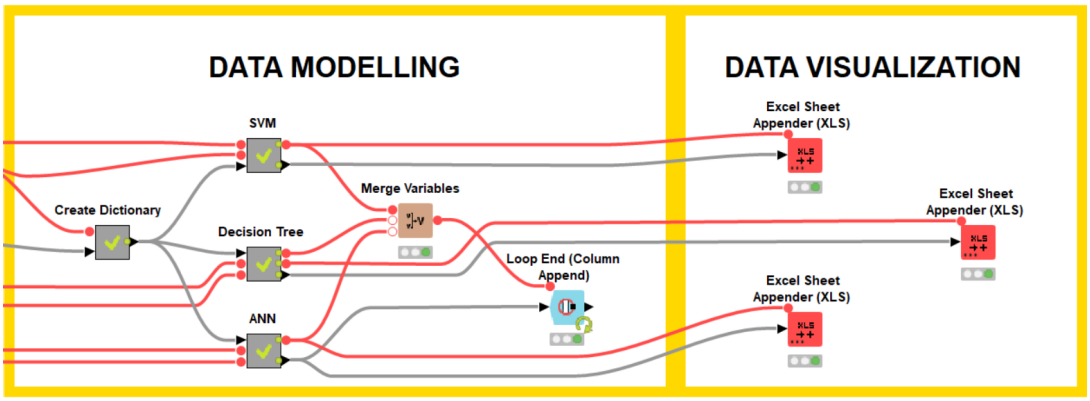


Figure 4.4: Workflow B - part 2: Classification of new data

4.1.3. Dashboard

The free version of Knime Analytics does not support the build of a dashboard or report. For this, we will use Qlik Sense. The output data from workflow B is saved and reopened in the Qlik Sense environment. Here, the new data is connected to the historic data used in the performance analysis in chapter 2. The data is connected such that information is shown based on characteristics of the corresponding system type, object name, or location. Settings for the level of detail can be chosen and an overview is automatically created. We will elaborate on this dashboard in chapter 6. A full representation is available in appendix B.

4.2. Training and validation set

The first step in modelling unstructured data is the preparation of the training and validation sets. In this section, we will first discuss the content of these data and the choices we made regarding assumptions and concessions for building the models. Next, we will discuss the labelling of these data with classification labels. The training and validation sets are partitions of one large data set.

4.2.1. Content of the training and validation set

As mentioned in the conclusion of chapter 2, the data on which the classification models are built are unstructured data. In practice, a Field Engineer is presented with a problem, solves the problem, and then writes a description of the problem, cause, and solution in a maintenance log. These three text blocks are the core of the training and validation sets. The sets are extended with a unique number, the Service Order number, to be able to connect results to the original data entry. In the data set, only corrective service orders (SOCs) are used because the problem description of preventive service orders (SOPs) is generally left blank or written as a single expression 'Technical Functional Maintenance'. SOCs that represent a second or third visit usually have a problem description that is equal to the original SOC. If this is not the case, the problem description of the last SOC is adapted such that it holds all available information about the problem. Only the SOCs with a solution are kept in the data set we only know the actual core problem if the problem has been solved. A second added parameter is the object name, which describes the type of system that is maintained. This parameter is added to be able to build separate models for each system type. A full representation of the input data is shown in Table 4.1.

Table 4.1: Transposed overview of the input data

Service Order No.	SOC17018003181
Object Name	EC-1 iVRI
Problem Text	iVRI meerdere keren weer uitgevallen, Melding: CPU C ID = o com fout VRI valt uit op gedoofd, resetten gaat alles op rood, staat nu op knipperen
Cause Text	Connector achterop de PSU is verbrand. Telefonisch contact gehad met klant
Solution Text	Medewerker heeft de PSU vervangen en de installatie regelt weer.
Class	Electrical failure

4.2.2. Assumptions and concessions of the training and validation set

The data set is filled with data from 2017 and extended with 2018 or 2019 data if more entries were required. 2017 is chosen because the performance analysis (section 2.3) showed the lowest second visit rate in 2017. We can therefore say that 2017 had the most accurate problem descriptions, assuming that each year the problems were equally difficult and solved with equal skill. 2018 and 2019 are chosen as extensions instead of for example 2016, because recent data is more similar to the problem descriptions that are written from this day forward. Changes in problem descriptions are among others due to change in personnel. Finally, the data set was filtered such that no empty cells remain and therefore full information is available.

The data set consists of 500 instances, evenly divided over ten system types (EC-1, EC-2, etc.). This amount is based on the research from Wei et al. (2017), who use batch sizes of 20 and 50 instances. Other researches (Edwards et al., 2008; Reshmy and Paulraj, 2015; Sen et al., 2019) prove that a larger data sets result in more accurate models, but we chose this amount to start with. We will reflect on the size of the data set in chapter 5.

4.2.3. Class labelling

As mentioned in section 3.3, the classification structure is a list of failure modes that are the class labels, which can be assigned to data entries. The creation of such a structure is based

on the variation of words available in the BoW of the training data set and the theory of failure modes (section 3.3.1). The goal is to create a class division as detailed as possible, without losing statistical significance on the size of subsets and correlation between the classes. A high distinction is therefore required. This step is a semi-automatic process which requires the knowledge of a maintenance expert (i.e. a Field Engineer). Because we are dealing with a proof of concept and the process of labelling training data is time consuming, the researcher has labelled the training data. The training data consists of 50 data entries of each of the ten different system types.

Table 4.2: Overview of possible failure modes used as class labels

Failure mode	**Description**	**Examples of failures**
Electrical failure	Failure of electrical components or wires	Burned fuse, melted cable sleeve
Mechanical failure	Failure of mechanical components	Defect push button on VRI, broken lock
Software failure	Error or bug in the software	Log error, boot error, bug
External damage	Damage caused by an external factor	Collision, cable gnawed by animal
Installation fault /human fault	Fault caused by wrong installation or repair of the system	Wrong components used, wrong boot sequence
No failure	No failure of the system	Client suspects a failure, but the system is up and running

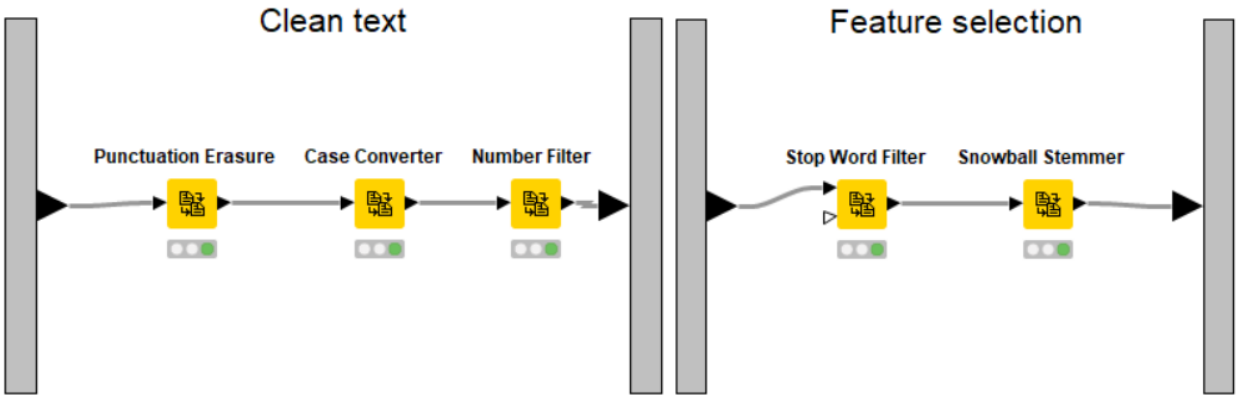
The chosen failure modes as shown in Table 4.2 have a high level of distinction. Next to general failure modes like electrical, mechanical and software failure, the BoW showed information on installation and/or human failures. Tinga (2013) creates a distinction between these two failure modes, but the data does not show this distinction well enough. On the other hand, the two groups together can be distinguished from the other failure types. Another failure type is external damage, which is caused by an external factor like an animal (e.g. a mouse who has gnawed through a cable) or a collision. Finally, we added a final option where no failure is present. This is possible when for example a client's monitoring system generates an error, but the traffic control system operates in fact without failure.

4.3. Data preparation

After completing the test data set, the data was transformed to proper input data on which a classification model can be built. This transformation includes cleaning the data, selecting the terms with which are most valuable to build a model on, calculating the term weights and transforming the text fields into document vectors. These vectors are the input data for building the classification models.

4.3.1. Data cleaning

The data cleaning step uses simple techniques like erasing punctuation, converting all cases to lower cases and filtering all numbered values from the data (Figure 4.5a). When we recall the earlier mentioned example ”System failed, system is switched off”, it is transformed to the form ”system failed system is switched off”.



(a) Cleaning text data (b) Selecting the most valuable features

Figure 4.5: Cleaning data and selection of the most valuable features

4.3.2. Feature selection

After cleaning the data, we filter the words that have no value out of the text fields. This process is also known as feature selection (Aggarwal and Zhai, 2012). Feature selection serves two main purposes. First, it makes training and applying a classifier more efficient by decreasing the size of the effective vocabulary. Second, feature selection often increases classification accuracy by eliminating irrelevant or noisy features. The goal is to remain with a small group of features (i.e. terms) that have a high value and are used often. An example of such a feature is the word 'damage'. This word is almost exclusively used to described the class label 'external damage'. For this research, we used a stop word filter and a stemming algorithm (Figure 4.5b). Other selection methods like manual selection and synonym lists can be added to improve the model's accuracy, but are left out of this research because of time restrictions.

For the stop word filter, we used a large filter, which removes words that are used often, but are unimportant to the meaning of a sentence. We chose a large stop word list, because we are looking at keywords that represent failure modes. When more sentiment is required in a model, a smaller stop word list is required that for example does not hold grammatical denials like 'not' or 'none'. The used Dutch stop word list, which was not available in the software, is created by Eikhart (2011) and available in appendix C. After this step, the example sentence becomes "system failed system switched off".

As stemming protocol, we used the Snowball stemming algorithm developed by Porter (2002). The Dutch version of this algorithm is based on Germanic language rules. This algorithm searches for common linguistical changes in words, like the use of accents on vowels (e.g. á, ë or ü) or the use of double letters in plurals (e.g. the Dutch word for bags changes from 'zakken' to 'zak'). The full algorithm is available in appendix D. The final form of the example becomes "system fail system switch off".

4.3.3. Defining term weights

With the text data cleaned and the proper features selected, we created the term weights (Figure 4.6). First, the sentences were parsed into single words (i.e. terms) and converted into a string variable. Next, we calculated the TF$\times$IDF values as term weights using Equation 3.2. As mentioned in section 3.2, term weights determine the placement of instances in a classification model. The TF$\times$IDF value is used as Kantardzic (2011) states that this is the most accurate term weight. It considers both the local term frequency in a document (TF) and the global document frequency compared to the number of documents that contain the considered term (IDF). The first word from the example, "system", occurs twice in the sentence (i.e. the document), so the Term Frequency is two. If that word occurs in no other

document and we have a total of five documents, the Inverse Document Frequency becomes $log(\frac{5}{1}) = 0.699$. The TF×IDF value becomes $2 \times 0.699 = 1.398$. In Figure 4.6, the TF and IFD values are calculated separately and simply multiplied.

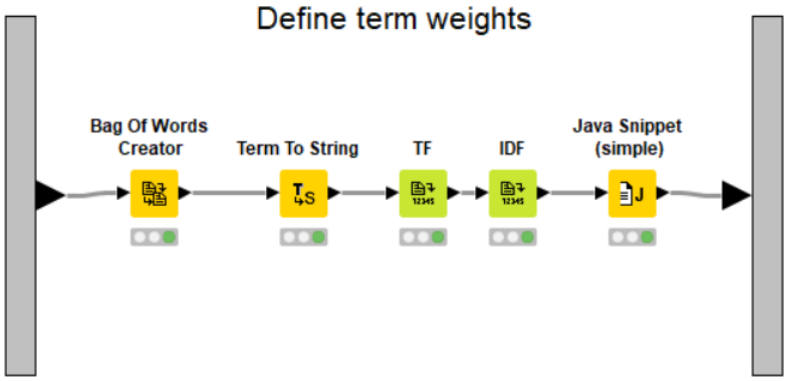


Figure 4.6: Defining term weight TFxIDF

4.3.4. TRANSFORMING DATA TO DOCUMENT VECTORS

The final step of preparing the text data is the conversion of terms into document vectors (Figure 4.7). The vectors consist of term weights. We have just calculated the term weight of the term "system", which is 1.398. As mentioned earlier, a higher number represents a more unique term. This must be repeated for each term, creating a list of vectors corresponding with a set of terms (Table 4.3). In addition, we linked the prepared class labels to the document vectors and each class label was marked with a colour for future reference. Finally, the document parameter (i.e. the full text) was removed from the data set as this parameter creates a distortion in the data modelling step.

Table 4.3: Example of document vectors

	system	**fail**	**switch**	**off**	**etc**	**...**
Vector 1	1.398	etc	...	...	...	...
Vector 2	etc	...	...	...	...	...
etc	...	...	...	...	...	...

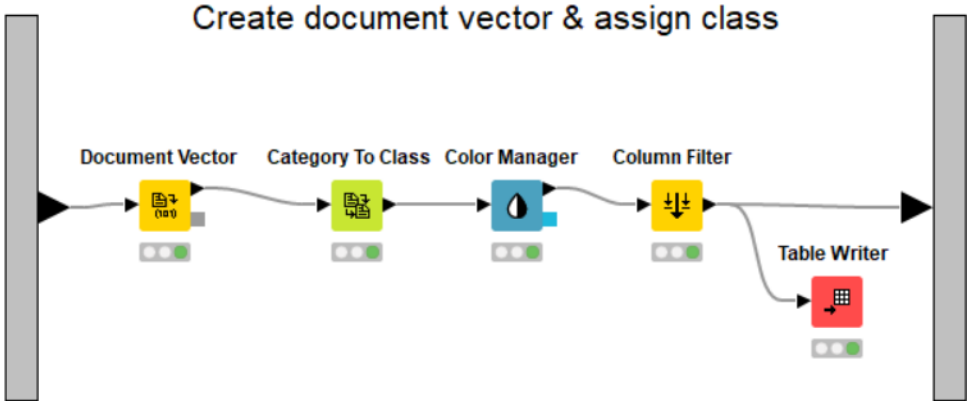


Figure 4.7: Creating document vectors

4.4. CLASSIFICATION ALGORITHMS

Knime Analytics offers complete algorithms for building and validating classification models. The build of the models is done through a learner node, which uses the document vectors from the test set as input data. The validation of the models is done through the predictor node, which uses the document vectors from the validation set and classifies the vectors based on the model. A scorer node is used to create a confusion matrix and calculate an accuracy score. In this section, we will successively discuss the decision tree, SVM, and ANN models.

4.4.1. DECISION TREE MODEL - C4.5

The decision tree algorithm we used is the C4.5 algorithm, as is explained in section 3.3. This is the algorithm on which the decision tree learner node is based. As shown in Figure 4.8, to train the learner node, first the data is partitioned. We chose a 75/25 percent partition ratio for the test and validation set, respectively. This ratio has proven to be most reliable in practical applications (Kantardzic, 2011).

Next, an optimization step is executed to find the optimal parameters of the learner node. Table 4.4 shows the parameters that are changed in the optimization loop. The parameters decide how the decision tree is built. The first parameter *minimum number of records* checks in each node of the tree whether the number of data entries left to split in that node is larger than a predefined minimum. If that is not the case, the last node becomes the leaf node. The step size of the values is set to one. The second parameter *quality measure* looks for either the largest information gain (Gain ratio) or the lowest chance of misclassification (Gini index). Recall from section 3.3 that the Gain ratio is determined by the difference in information entropy between the current node and the possible nodes downstream. Information entropy can be defined as the degree of disorder in a spatial area. The goal is to find a downstream node with the lowest possible entropy. The Gini index is the probability of a sample being misclassified if we randomly pick a label regarding the distribution in a branch. The third parameter *pruning method* is used when the decision tree is fully built. This parameter decides whether or not each branch of the tree holds valuable information. This is done with the Minimal Description Length (MDL) method. Branches are cut (i.e. pruned) when the information value is low.

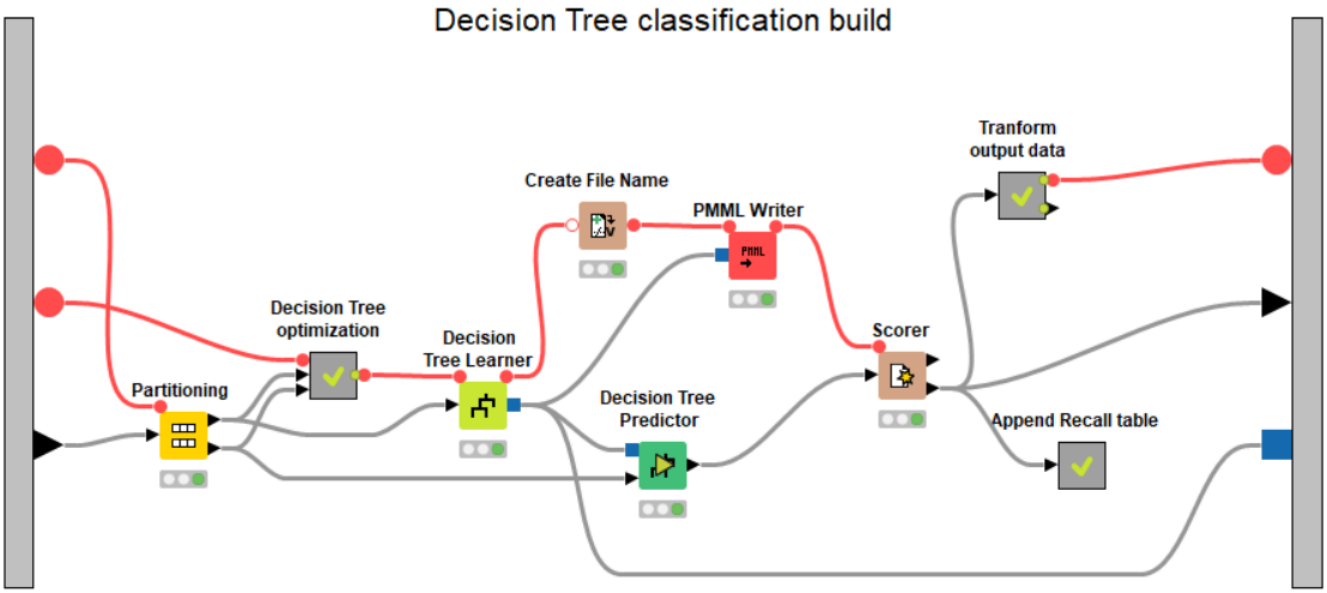


Figure 4.8: Building the Decision Tree classification model

The optimization loop uses a brute force method that runs through all possible value combinations. This is possible due to the low number (56) of combinations, which is the multiplication of the number of possible values per parameter. The best parameter values

are saved and the optimal decision tree model is rebuild. The model is also saved as a Personalized Print Markup Language (PPML) file, which is available for use in the workflow for new data entries (Workflow B). A PMML file is a generally used file extension for data models. In parallel, the model is validated with the validation set. Finally the accuracy score is calculated and the results of all loops (i.e. all system types) are transformed to the required form of output data. Next to the overall accuracy score, the score per class label is also saved. This score is called 'recall'. We will use the recall to determine how difficult it is to accurately classify each failure mode.

Table 4.4: Decision tree tuning parameters

Parameter	Values	Description
Minimum number of records	2,...,15	Minimum number of data entries required to branch a node in the tree
Quality measure	Gain ratio, Gini index	Measurement method for selecting the best input feature in each iteration
Pruning method	No pruning, MDL	Method for simplifying the decision tree and preventing over- or underfitting

4.4.2. SVM MODEL - SEQUENTIAL MINIMAL OPTIMIZATION

The SVM algorithm we used is the SMO algorithm, as explained in section 3.3. This is the algorithm on which the SVM learner node is based. The build of the SVM model is similar to the decision tree. The only differences are the learner and predictor nodes, which are now for an SVM model, and the optimization of the SVM learner node. Table 4.5 shows the parameters that are optimized in this optimization loop. All parameters have a step size of 0.1.

The first and most decisive parameter is the *overlapping penalty*, also known as the C-parameters. As explained in section 3.3, the overlapping penalty determines a trade-off between the linearity of the decision boundaries and the number of misclassified instances. A high linearity provides a simple model to which many data sets are applicable, but could resolve in misclassified instances. A high overlapping penalty corresponds with a high linearity.

The second parameter is the *type of kernel*. The use of a kernel is, as explained in section 3.3, to use linear separators for a non-linear problem. In other words, the kernel function is a shortcut to do certain calculations faster which otherwise would involve computations in a higher dimensional space. Knime Analytics support three kernel types. After a series of tests, the Polynomial kernel proved to be most suitable for our data set. As the name already explained, this kernel looks for a polynomial function which separates the classes in the search space. Figure 4.9 shows several forms of the polynomial function.

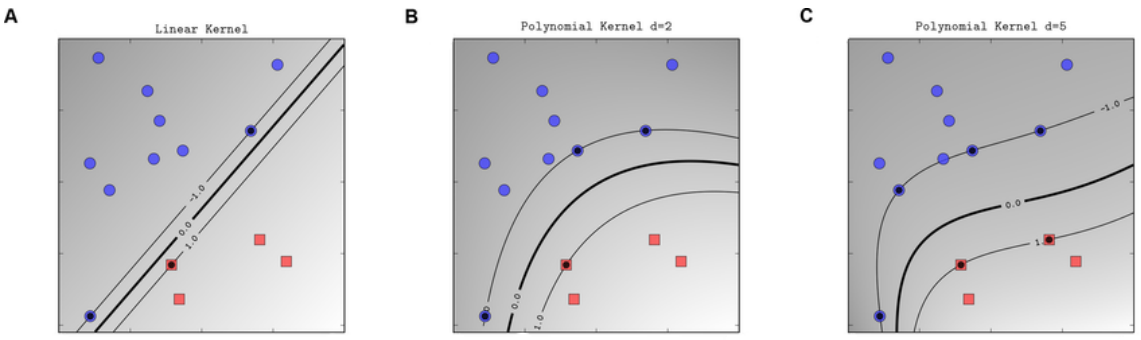


Figure 4.9: Examples of a polynomial kernel function with different power [d] values

The equation of the Polynomial kernel is similar to the example kernel (Equation 3.12) explained in section 3.3:

$$K(x, y) = (\gamma(x^T y) + b)^d \tag{4.1}$$

where $K(x, y)$ is the kernel and x and y are n-dimensional inputs (i.e. the document vectors). d is the power, which changes the curvature of the equation. b is the bias, also known as the offset, which changes the position of the equation. γ is the scaling factor of the equation. x and y are known values, determined when creating document vectors. Often, the offset b and scaling factor γ are set to one as tuning just the power d creates a proper separation between classes. In this research, we have let the optimization loop use a random search method that chooses up to 500 different value combinations of the overlapping value, power, offset, and scaling factor (Table 4.5). This is required due to the high number of combinations, which is the multiplication of the number of possible values per parameter. The ranges of the values are determined by fixing all but one parameter and observing the influence of the free parameter on the objective value.

Table 4.5: Support Vector Machine tuning parameters

Parameter		**Values**
Overlapping penalty		1.0, ..., 6.0
Kernel - Polynomial	power	1.0, ..., 3.0
	offset	0.1, ..., 2.0
	gamma	0.1, ..., 2.0

4.4.3. ANN MODEL - RESILIENT PROPAGATION

The ANN algorithm we used is the RPROP algorithm, as explained in section 3.3. This is the algorithm on which the ANN learner node is based. The build of the ANN model is just as the SVM model similar to the decision tree. The only differences are the learner and predictor nodes, which are now for an ANN model, and the optimization of the ANN learner node. Table 4.6 shows the parameters that are optimized in this optimization loop.

The first parameter is the *number of iterations*. As explained in section 3.3, a neural network is iterated to lower the difference between the calculated output value and the actual output value, also known as the error. The second parameter is the *number of neurons in the hidden layer*. Neurons are part of the hidden layer. Each neuron transforms a set of input values to an influence on the output value. The ranges of the values are determined by fixing one parameter and observing the influence of the free parameter on the objective value.

Table 4.6: Artificial Neural Network tuning parameters

Parameter	**Values**	**Step size**
Max number of iterations	100, ..., 1000	50
Number of neurons in the hidden layer	10, ..., 100	5

4.5. Model validation

As described by Law (2015), validation is the process of determining whether a simulation model is an accurate representation of the system, for the particular objectives of the study. A simulation model is valid when the accuracy level is higher than a predetermined threshold. For this research, the accuracy threshold cannot be established as no previous model exists for the same data set. Therefore, we will not compare the accuracy score with a threshold, but we will reflect on two studies from literature that are most similar to this research (Edwards et al., 2008; Wei et al., 2017). Both studies classify text data from malfunction reports. Edwards et al. (2008) classified on the labels *corrective* and *preventive* maintenance. The constructed decision tree and neural network models had an accuracy of respectively 85% and 83%. The used data set had 842 records. Wei et al. (2017) classified on five failure modes: *mechanical failure, electrical failure, human failure, external environment,* and *secondary equipment fault*. The proposed model, a Recurring Neural Network (RNN), tested small batch sizes of 10, 20, and 50 records. The model was able to reach a maximum accuracy of 64%, which was compared with a threshold value of 50%. We will reflect on these studies with the accuracy results from our models which consist of a batch size of 50 records per model and have six failure modes.

In our model, the accuracy score is determined by running a validation set through the model and scoring the model based on the number of false negative, false positive, true negative, and true positive classifications. The results are summarized in a confusion matrix and the accuracy score is calculated as follows:

$$\text{Accuracy} = \frac{\text{Number of true positives}}{\text{Total number of records in validation set}} \tag{4.2}$$

4.6. New data classification

With the classification models built, new data can be classified. As mentioned in section 4.1, new data runs through workflow B. This workflow is very similar to workflow A. Therefore, in this section we will only discuss the main differences.

The first difference is the use of a dictionary. To improve the speed of the model we chose to create a dictionary solely based on the words from the created model and words from the new data set. The dictionary is a collection of all possible terms. A dictionary is created for each system type separately and used when the PMML model for this system type is requested. We can also run larger data sets through the model that are not partitioned on system type, but for example system group (i.e. highway and intersection systems). Then, the dictionaries will be larger as well.

The second difference is the use of the predictor node (Figure 4.10). The trained model from workflow A is called by a PPML reader node. The new data workflow handles each system type separately and loads the corresponding training model. Then the new data set is classified by the predictor node and prepared for the data visualization step. Partial classification data is added to understand how certain a classification is (e.g. 30% of 60%).

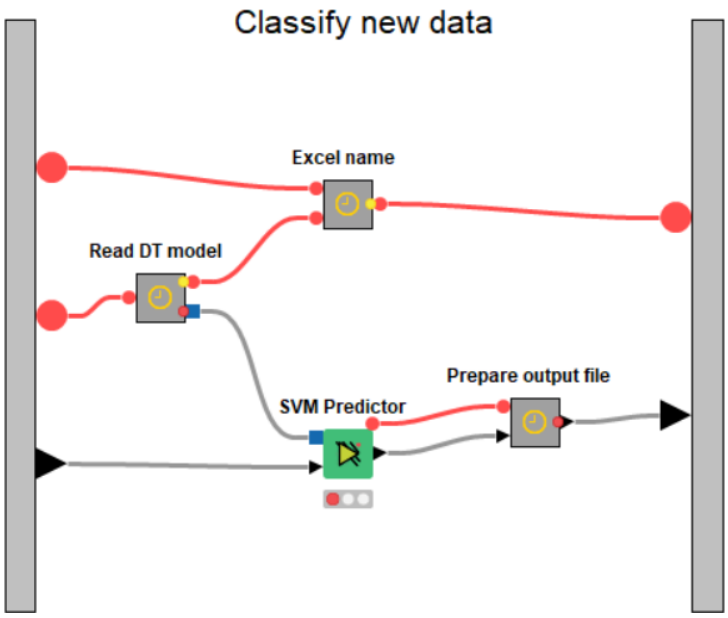


Figure 4.10: New data classification

The third difference is the data visualization. Knime Analytics does not support the build of a dashboard. Therefore, the result is collected in an Excel file which can be used in the dashboard made with Qlik Sense. This Excel file holds the parameters *Service Order Number*, *System Type*, *Problem Text*, *Probability class label A*, ..., *Probability class label F* and *Assigned Class*.

4.7. Conclusions

In this chapter, we discussed the design of a classification model, which gives answer to the third research question: "How can data be modelled to provide guidelines for optimizing the maintenance strategy?". The designed model can be used to assign class labels to new service orders. The class labels provide information on the possible failure mode. This information can be used to help a Field Engineer to find a problem solving approach for the service order.

Next to answering the research question, we can draw some conclusions about the model based on its design. We made come concessions on the size and quality of the data set and the way we selected the features. These concessions are made because of time restrictions and the availability of feature selection methods in Knime Analytics. The concessions will have a negative influence on the accuracy score, which we will elaborate on in chapter 5.

Also, we excluded the analysis of terms that consist of n-grams like bigram and trigrams. To include this properly, the test and validation sets for the training model must be extended significantly because the occurrence of an n-gram decrease when n increases. The added value of n-grams is unknown, but could be considered as a recommendation.

Finally, the validation of the training model is measured by the accuracy of the classification of data. The threshold for this accuracy that reflects on how well the model is performing cannot be objectively determined as there is no known model available that has been used for our data set. Therefore, the results will be compared to two known studies that are similar to this research.

5. Model Evaluation

In this chapter, we will evaluate the designed classification model from chapter 4. Doing so, we will answer the fourth research question: ”How can the results from the data model be used to improve the maintenance strategy?”. To answer this question, first, we will evaluate the model validation results (i.e. the accuracy scores) in section 5.1. Then, in section 5.3 we will discuss what parts of the model highly influence the accuracy score. Next, in section 5.4 we will discuss the limitations of the model and we will end this chapter with conclusions in section 5.5.

5.1. Model accuracy

As mentioned in chapter 4, the performance of the model is defined by the percentage of correctly classified instances from a validation set based on a model built with a test set. This is also known as the accuracy score. In this section, first we will discuss the accuracy score of each system type separately, which is the results of the designed model. Then, we will compare these results with the same data set, but partitioned on system groups (i.e. highway and intersection systems) instead of system types. We will repeat this for the data set with all systems combined. This will give insight in whether the system types are maintained similarly or not. The goal of this section is to determine which classification algorithm is most accurate, whether the accuracy score is good, and what the best partition of the data set is.

5.1.1. Accuracy score - system type

We used the full data set consisting of 500 instances as proposed in section 4.2. We partitioned the data set in ten subsets, defined by the system types (Table 5.1). Each subset has a different model and therefore a separate accuracy score. The subsets contain data from 2017 which was extended with 2018 and 2019 if the required number of instances (50) was not reached yet. For each system type a separate SVM, decision tree, and ANN model were created and validated. Each subset of 50 instances was split into a test and validation set with a 75/25% partition to 37 and 13 instances, respectively. The partition is done by the method of stratified sampling, which means the data is split based on the assigned class labels and the partition ratio. For example, given that ten of the 50 instances are labelled ’electrical failure’, seven are part of the test set and three are part of the validation set.

Table 5.1: Division of instances - system type partition

Partition	Instances	Partition	Instances
EC-1	50	EC-2.5 iVRI	50
EC-1 iVRI	50	OS-0	50
EC-2	50	OS-2	50
EC-2 iVRi	50	OS-5	50
EC-2.5	50	OS-6	50

To lower the statistical error of the accuracy scores, a total of 40 repeated runs are executed. This means that each model is build 40 times with a different random number each time. 40 runs is enough to lower the statistical error to less than 5% for each system type. In each run, the random number (i.e. seed number) for splitting the data set into a test and validation set is changed. A seed number is a number used to initialize a random number generator. We use

seed numbers to be able to recreate our results. Each run has an accuracy score of which we take the average. So, given system type EC-1, 40 runs are executed with seed numbers 1, 2, ..., 40. Each run has an accuracy score (e.g. 50, 55, or 60%) and the average is used to compare the system types and the classification algorithms with each other. The results of the runs are collected in Table 5.2. In the lower part of the table, the averages over the system groups and all systems combined are shown. We will compare these averages with the results from the second and third test where the data set is partitioned on system groups and all systems, respectively. The optimal parameters for each model are discussed in section 5.2.

Table 5.2: Accuracy scores per system type for the DT, SVM, and ANN models

System type	**Accuracy score DT**	**Accuracy score SVM**	**Accuracy score ANN**
EC-1	53%	58%	58%
EC-1 iVRI	60%	63%	63%
EC-2	39%	58%	62%
EC-2 iVRi	49%	63%	63%
EC-2.5	54%	53%	49%
EC-2.5 iVRI	43%	50%	55%
OS-0	66%	71%	75%
OS-2	68%	69%	68%
OS-5	61%	61%	55%
OS-6	57%	64%	61%
Average Highway systems	63%	66%	65%
Average Intersection systems	50%	57%	58%
Average All systems	55%	61%	61%

We can observe several facts from the results. The first fact is that the ANN and SVM model have a better average score (61% for both models) over all the system types, followed by the DT model with 55%. Even though the ranges (i.e. difference between maximum and minimum scores) of the accuracy scores do fluctuate a lot with 21% for SVM, 26% for ANN, and 29% for DT, we can observe that the SVM and ANN models perform almost always at least as good as the DT model. A deviation can be observed for the EC-2.5 and OS-5. The second fact is that the highway (OS) systems are much easier to predict than the intersection (EC) systems. The highway systems have an average score of 63, 66, and 65% for the DT, SVM, and ANN models, respectively. The intersections have a lower average score of 50, 57, and 58% for the DT, SVM, and ANN models, respectively. The third fact is that the ANN and SVM models score better than the research from Wei et al. (2017), discussed in section 4.5, which has a maximum accuracy of 64%. This complies with our expectation as the data sets of our study and the reference article have the same size and roughly the same number of class labels. To establish whether the partition on system types provides the best results for this data set, we will compare the results with two larger partitions, system groups (i.e. highway and intersection systems) and all systems together.

5.1.2. Accuracy score - system group

For the system groups, we used a partial data set consisting of 400 instances. Of the ten system types, six are intersection systems and four are highway systems. We used 200

instances of each system group, evenly divided over the available system types (Table 5.3) with a separate score per system group. This creates batch sizes of four times as large as the test on system types, which should result in a higher accuracy score if the system types in the groups (e.g. EC-1, ..., EC-2.5 iVRI in the group 'Intersection systems') have similar failure modes. In other words, if models on system type level are compatible, they can be combined to create a larger dataset on system group level.

Table 5.3: Division of instances - system group partition

Partition	**System types in partition**	**Instances**
Intersection systems	EC-1, EC-1 iVRI, EC-2, EC-2 iVRi, EC-2.5, EC-2.5 iVRI	200
Highway systems	OS-0, OS-2, OS-5, OS-6	200

This data set was run 40 times again. The results are shown in Table 5.4. We can observe an accuracy score that again shows that the ANN (53%) and SVM (54%) models are more accurate than the DT model (49%). Also, the accuracy scores are much lower than the averages taken from the previous test with a partition on system types (Table 5.5). In general, a larger batch size (i.e. a larger test and validation set) will resolve in a more accurate training model. In that case, the partition on system groups should have a higher accuracy than the partition on system types. This is not the case and we can therefore conclude that the division between system types is too large to combine them as system groups.

Table 5.4: Accuracy scores per system group

System type	**Accuracy score DT**	**Accuracy score SVM**	**Accuracy score ANN**
Highway	53%	55%	54%
Intersection	45%	53%	52%
Average All systems	49%	54%	53%

Table 5.5: Comparison table - Averages of accuracy scores per system type

System type	**Accuracy score DT**	**Accuracy score SVM**	**Accuracy score ANN**
Highway	63%	66%	65%
Intersection	50%	57%	58%
Average All systems	55%	61%	61%

5.1.3. ACCURACY SCORE - FULL DATA SET

To substantiate our conclusion, we will also look at all systems together. The results, according to our last conclusion, should result in a lower accuracy score than the test with a partition on system types. For the test with all data in one group, we used the full data set consisting of 500 instances. All instances are evenly divided over ten system types (Table 5.6) with one accuracy score for the full data set.

Table 5.6: Division of instances - full data set

Partition	System types in partition	Instances
Full data set	EC-1, EC-1 iVRI, EC-2, EC-2 iVRi, EC-2.5, EC-2.5 iVRI, OS-0, OS-2, OS-5, OS-6	500

This data set is run 40 times again. The results are shown in Table 5.7. We can observe from this Table that again the SVM model (52%) is more accurate than the DT model (49%), but this time the performance of the ANN model (50%) is closer to the DT model. Also, the scores are again lower than the partition on system types (Table 5.8), which confirms the conclusion that the best partition of the data set is on system types.

Table 5.7: Accuracy scores for all systems

System type	Accuracy score DT	Accuracy score SVM	Accuracy score ANN
All systems	49%	52%	50%

Table 5.8: Comparison table - Averages of accuracy scores per system type

System type	Accuracy score SVM	Accuracy score DT	Accuracy score ANN
All systems	55%	61%	61%

5.2. Optimal model parameters

As explained in section 4.4, each model has its own optimal parameters. The optimal parameters fluctuate between system types but optimal ranges can be observed. Tables 5.9, 5.10, and 5.11 hold the optimal parameters for the DT, SVM, and ANN models, respectively.

Table 5.9: Optimal decision tree tuning parameters

Parameter	Optimal values
Minimum number of records	2, 3, or 4
Quality measure	Gini index
Pruning method	No pruning

Table 5.10: Optimal Support Vector Machine tuning parameters

Parameter		Optimal values
Overlapping penalty		1
Kernel - Polynomial	power	2.4
	offset	0.7
	gamma	1.9

Table 5.11: Optimal Artificial Neural Network tuning parameters

Parameter	Optimal values
Max number of iterations	100, 150, or 200
Number of neurons in the hidden layer	15, 20, or 25

5.3. Model sensitivity

During the evaluation of the model, several aspects were observed that influence the model's performance. A lower performance is defined as a lower accuracy and lead to a lower reliability of the model. In this section, we will discuss these aspects. First, we will discuss the size of the data sets and the quality of the input data. Then, we will discuss which class labels are easiest to classify and finally we will discuss how more extensive input data influences the reliability of partial classification.

5.3.1. Size of the data set

In section 4.5, we discussed two reference articles (Edwards et al., 2008; Wei et al., 2017). Both the input data and the result from our model comply with the research from Wei et al. (2017). The reference article used a Recurrent Neural Network (RNN) instead of an ANN, which is a more complex neural network and should yield a higher accuracy. When we compare our research to the other article (Edwards et al., 2008), we can observe that our data set is roughly fifteen times smaller and uses three times as many class labels. The accuracy score from Edwards et al. (2008) is around 85% for a (unknown) decision tree classifier and 83% for a (unknown) neural network. The difference in accuracy can be related to the chance of misclassification, which becomes higher when more class labels are used. Another difference is that a larger data set holds more examples of failure modes, which results in a better informed model. Recall from section 3.2.2 that a better informed model has a larger diversity of the document vectors, which leads to a better separation of the classes in the spatial area.

To proof whether a larger data set creates a significant increase of the accuracy score, we extended the data set of three system types (EC-1, EC-2, and OS-0) by 50 instances to 100 instances. We ran the data set through the model for 50, 75, and 100 instances. The resulting accuracy scores are collected in Figure 5.1. When we look at the trend between data sizes for each system type, we can observe that the accuracy scores do not significantly increase with the increase of the size of the data set. Only system type EC-1 shows a steady increase. This contradicts the hypothesis, which is based on literature (Aggarwal and Zhai, 2012; Larose, 2005), that the accuracy score would increase with the increase of the size of the data set. A possible explanation for this contradiction is that the model currently has several limitations that influence the performance. We will look at these limitations in section 5.4. After improving these limitations, this test on data size should be repeated.

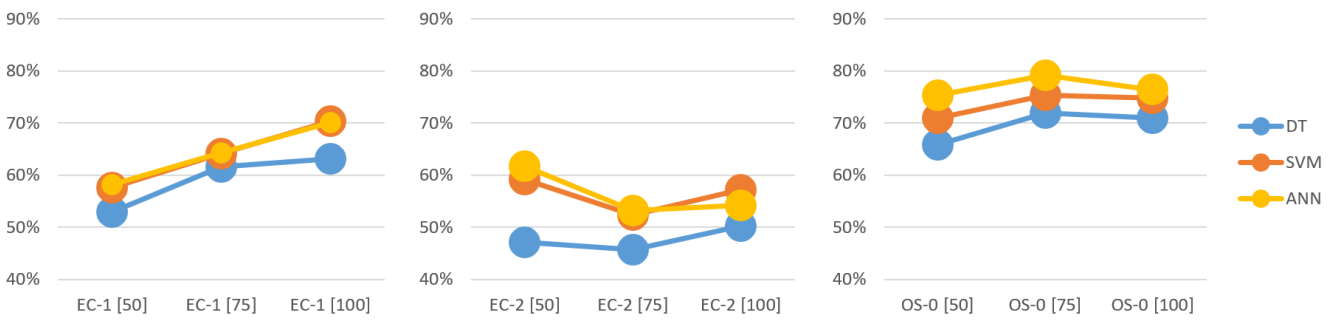


Figure 5.1: Accuracy scores per data set size per system type

5.3.2. Quality of the input data

Closely related to the size of the data set is the quality of the input data. This quality is influenced by two human actions. The first action comes from the person who describes the problem. This is often a client, who has less knowledge about the system than a Field Engineer. Also, the client has often only partial access to the monitoring systems. This could results in a wrong observation of the problem or a low detailed description. The second action comes from the Service Desk employee, who is responsible for assessing the problem and consulting a Field Engineer when required. The Service Desk employee enters the problem description in the ERP system. These two human actions are a large factor on the amount of data and detail of the description. Naturally, a description such as 'System is down' is not enough information to separate electrical failures from software failures. To elaborate on this issue, we will discuss an example of the influence of a detailed problem description on the result of partial classification in the next section.

5.3.3. Partial classification

Each instance in a data set is classified by the classification model. This classification can be partial, as mentioned in section 3.2.2. What we want to observe is the influence of the input data (i.e. the problem description) on the partial classification. For instance, does the model provide us with the class label 'software failure' with a partial match of 25% or 60%. This influences the confidence we have in the outcome of the model.

To observe this influence, we used an example sentence and extended it step by step. Figure 5.2 shows us six versions of the sentence with the partial matches to four of the six class labels. The match to the remaining two class labels is zero. We can observe that the smallest sentence has a highest partial match of 35% with the label 'software failure'. This match changes with the addition of more information and to an 78% match with the label 'electrical failure'. We can conclude from this example that the influence of a detailed problem description is high.

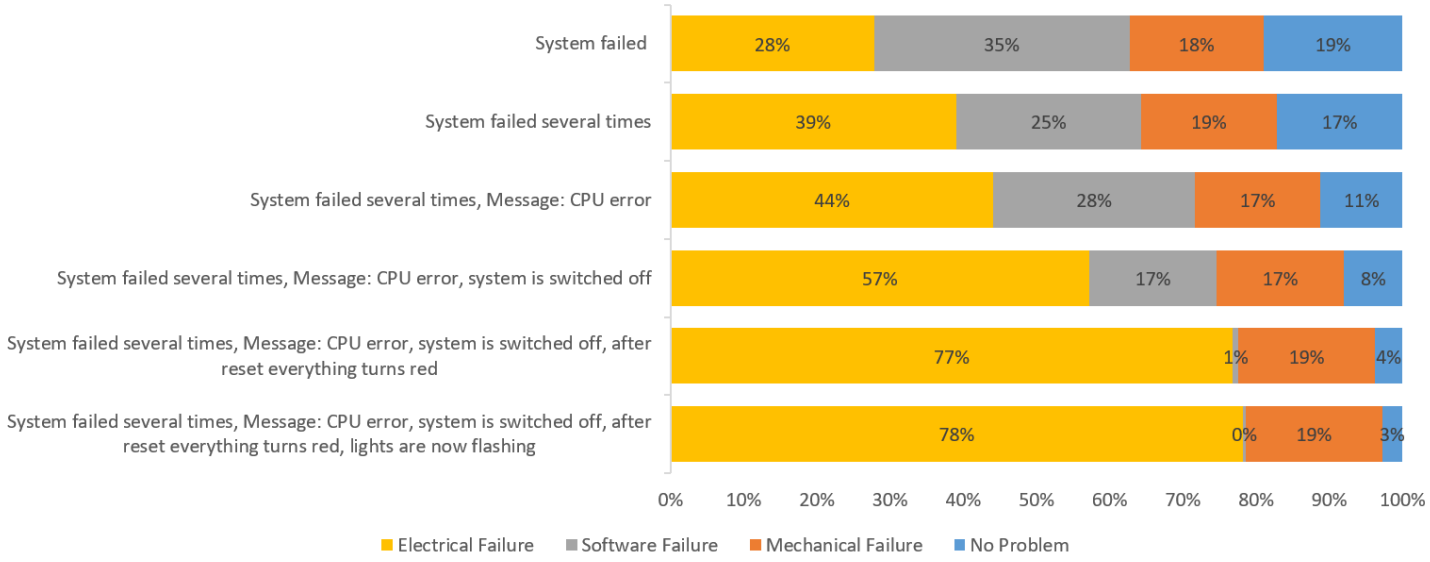


Figure 5.2: Influence of problem description detail on partial classification accuracy

5.3.4. Class label accuracy

Next to the influence of detailed input data on the partial match, the question rises on how well each label can be predicted. To measure this, we looked at the number of correctly classified instances per class label, which can also be described as the accuracy per class label. We used the partition on system types as this partition resulted in the best accuracy scores. Figure 5.3 shows the average percentages of correctly classified instances.

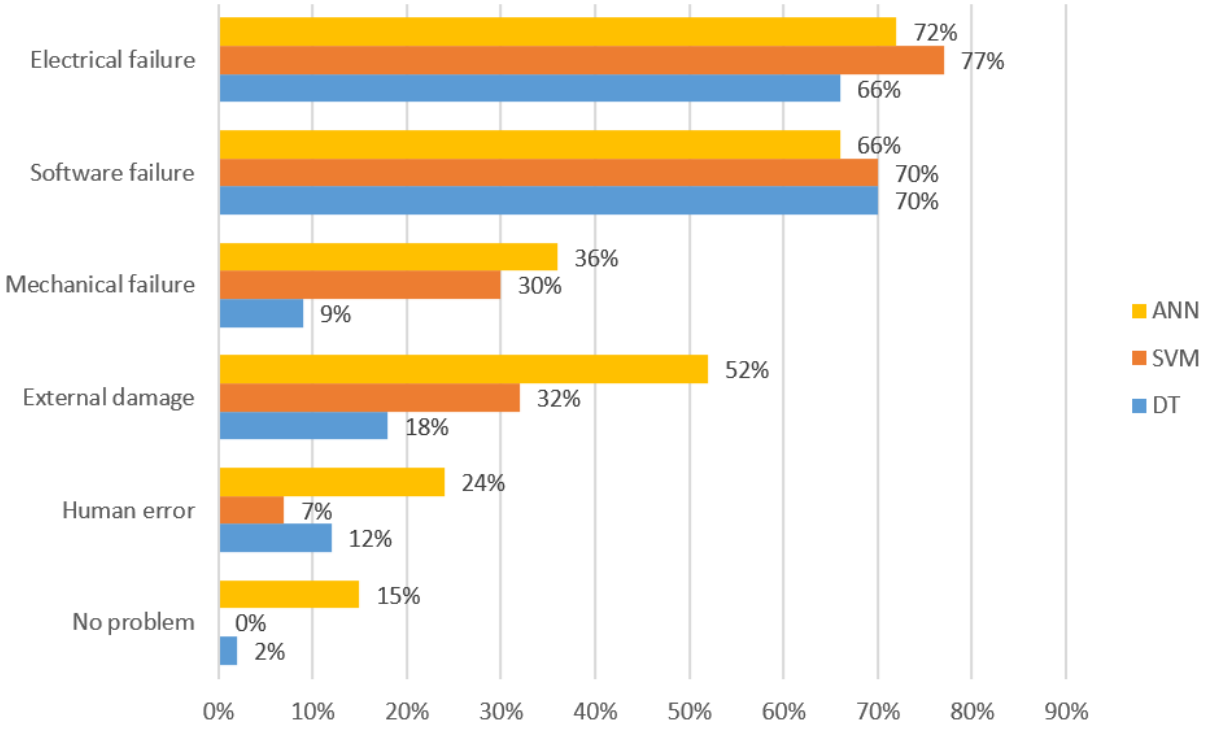


Figure 5.3: Accuracy scores per class label

We can observe a large difference in the accuracy scores of each class label. The labels *electrical failure* and *software failure* are relatively easy to predict for all models, *mechanical failure* and *external damage* are only moderately easy to predict for the ANN model, and *human error* and *no problem* are very difficult to predict for all models.

To look at the incorrectly classified labels in more detail, we created a confusion matrix (Table 5.12) that shows what percentage of an original class label is predicted by the model as that same or another class label. For example, 24% of all Service Orders with the class label *electrical failure* are classified by the model as a software failure. A full overview per model type (ANN, SVM, and DT) is available in appendix E. We can observe from Table 5.12 that both *electrical failure* and *software failure* are relatively easy to predict. *Mechanical failure, external damage,* and *human error* all have some distinctive value as they are partially correctly predicted, but are also often seen as *electrical failure* or *software failure*. *No problem* is almost impossible to predict correctly.

Table 5.12: Confusion matrix of actual (left) and predicted (top) failure modes

	Electrical failure	**Software failure**	**Mechanical failure**	**External damage**	**Human error**	**No problem**
Electrical failure	72%	24%	0%	1%	1%	2%
Software failure	26%	69%	0%	1%	1%	2%
Mechanical failure	32%	34%	25%	2%	1%	5%
External damage	36%	24%	1%	34%	0%	5%
Human error	42%	41%	1%	1%	14%	1%
No problem	49%	37%	1%	7%	0%	5%

The results are partially caused by the large number of Service Orders with the class label *electrical failure* or *software failure* in comparison to the other labels. These types of failure modes occur more often and the data set therefore holds more information on these labels. Also, the class labels *external damage* and *human error* are hard to predict, because these types of failures often result in an electrical or software failure. For example, a wire is connected to the wrong port which creates a shortcut. Then, at first an electrical failure is observed, which will only be assigned the label *human error* after further investigation. Finally, the label *no problem* is hard to predict, because this event occurs very seldom and is by definition not a type of failure mode. From the problem descriptions of these events, we can observe the general opinion that these events are often caused by small drops in the electrical grid which force the systems to give a non-fatal error and automatically reboot. In general, the class label accuracy results require a reassessment of the class labels. This should be done in cooperation with an expert group of at least a Field Engineer, Software Engineer, Hardware Engineer, and Maintenance Engineer.

5.4. MODEL LIMITATIONS

Next to the sensitivity of the data set, we observed several limitations to the model. The main limitation is the selection of features. Currently, only a stop word list and stemming algorithm are used to reduce the number of terms. Additional steps like a synonym list and manual selection could further decrease the number of terms. A smaller set of terms will result in a more accurate and faster model.

The second limitation is the stemming algorithm. For our model, we used the Snowball stemming algorithm by Porter (2002). The algorithm tends to truncate words more often than it finds the actual stem. For example, the Dutch word for 'making' has forms like 'maken' and 'gemaakt'. These two forms are truncated to 'mak' and 'makt', while the actual stem is 'maak'. In the model, these differences become separate terms and therefore get a higher term value as the terms become more unique. In addition to the error rate, the Snowball stemmer does not support a tag feature. Such a feature, often called a Part-Of-Speech (POS) tag, can add a label to a term that tells what type of word it is (e.g. verb, noun, or adjective). This can for example be useful when only keywords are allowed in a classification model.
The third limitation is the use of free software. This has caused the need for multiple tools that currently are only linked through CSV files. Also, there is no direct link to the ERP system. Many human interventions are required to operate the model, which increases the chance of errors and lowers the user friendliness. We will elaborate on this matter in the implementation plan (chapter 6).

The fourth limitation is the depth of the class labels. As explained in section 3.3, we have chosen types of failure modes as class labels and we have chosen a high level. This high level has as an advantage that distinctions between classes can easily be made. The downside of that advantage is that an electrical failure can still have many underlying failure modes. An expert opinion is required to reassess the class labels. The effect of more detailed classes cannot be measured with the current data set. More detailed classes should result in a higher accuracy, but only if the selected features are distinctive enough.

The final limitation is the use of only single words as terms. Groups of words, for example bigrams or trigrams, are more distinctive. Due to the higher distinction, a larger data set is required. When used properly, groups of words should result in a higher accuracy score.

5.5. Conclusions

In this chapter, we discussed the evaluation of the model we designed in chapter 4. This evaluation gives answer to the fourth research question: "How can the results from the data model be used to improve the maintenance strategy?". We will provide the answer with the following conclusions.

First of all, the evaluation of the model shows that it is possible to extract information from text data and use it for improvement of a maintenance strategy. The model is only a proof of concept, which needs several improvements before it can add real value to the maintenance activities on a daily basis, but it adds to our knowledge on what information is in the data set and how we can use that information. This answers several of the questions that lead to our core research problem.

Furthermore, the evaluation of the accuracy score has shown that the model is most suitable for a data set partitioned on system types. We can therefore conclude that the system types show a level of distinction which is too high to combine the data sets. The evaluation has also shown that the ANN model is more accurate than the SVM and DT models. As an ANN is only a simple neural network, we can conclude that neural network classifiers are most suitable to our data set. It also implies that classifiers based on decision rules are less accurate than classifiers based on a vector space model for our data set.

The observed accuracy scores are in line with the reference articles chosen in section 4.5. From this, we can conclude that the model performs well. At the same time, we can conclude from the observation of the influence of the problem description on the partial match that a detailed problem description highly influences the match percentage. In addition, we have seen that not all class labels are easy to predict. Especially the classes *human error* and *no problem* showed low accuracy scores. A third influence on the accuracy score that we have observed is the size of the data set. This complies with our knowledge from literature.

Next to the performance effects, we have observed multiple limitations to the model that require improvement. These limitations are the use of only a few feature selection tools, a moderately accurate stemming algorithm, the lack of a connection between the tools and the ERP system, the level of detail of the class labels, and the use of only single words as terms. After improvement of these limitations, the model can be implemented and improved, which we will discuss further in chapter 6.

6. IMPLEMENTATION

In this chapter, we will discuss the required steps for implementation of the classification model. This chapter will therefore answer the fifth research question: ”How can the data model be implemented in the maintenance strategy?”. First, we will discuss the implementation of the designed classification model in section 6.1 and then the recommendations for further development of the model in section 6.2.

6.1. IMPLEMENTATION OF THE DESIGNED MODEL

The classification model from chapter 4 is built as a proof of concept. Before implementing this model in the daily work routine, the model's performance must be increased by addressing most of the limitations from section 5.4. Figure 6.1 shows the workflow that should be used as a guideline for implementation. We will first discuss the required improvement steps and then the integration of the model with the ERP system. We will end this section with a rough timeline for the implementation.

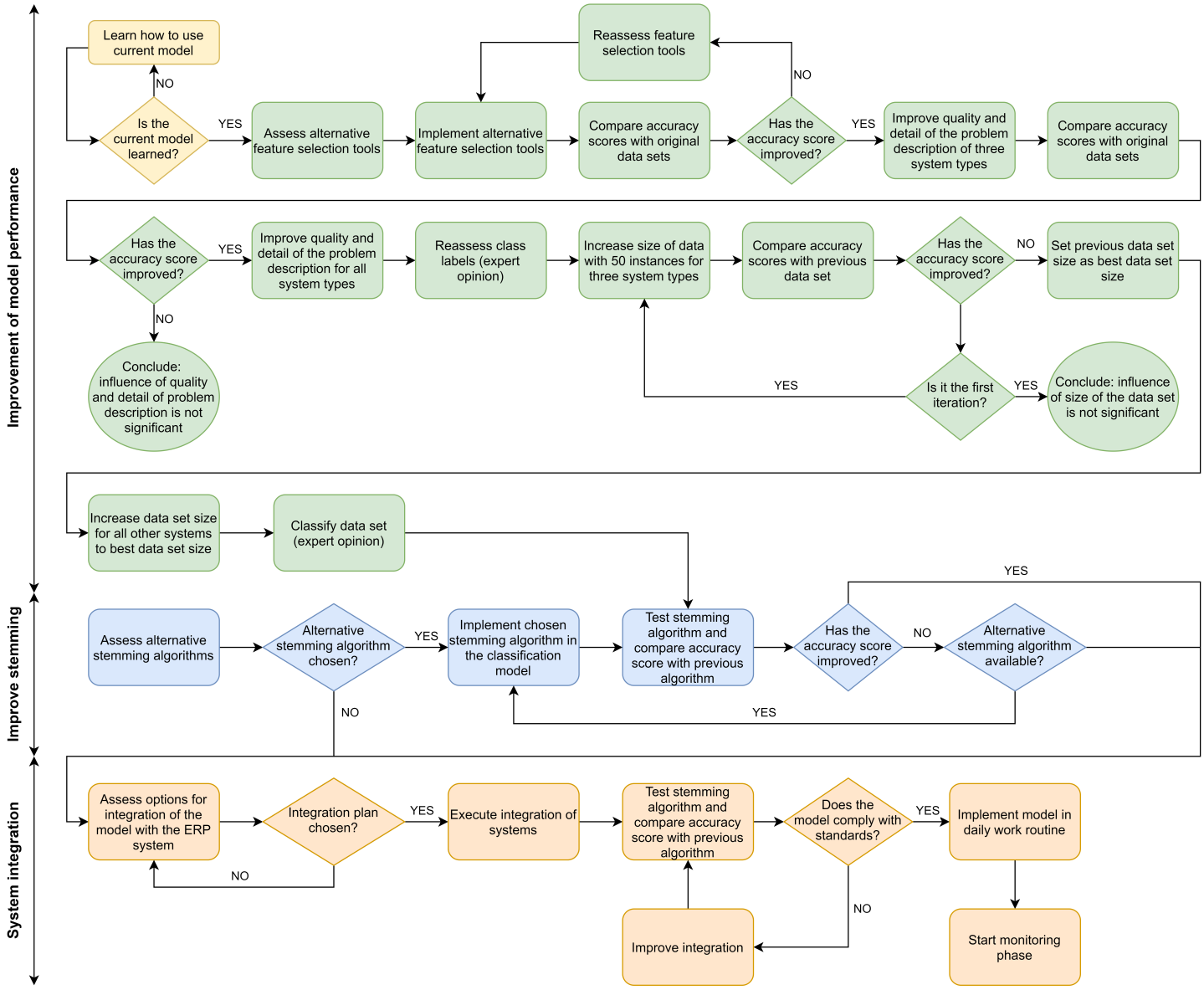


Figure 6.1: Workflow for implementation of the current classification model

6.1.1. Assessment of alternative feature selection tools

The first step in improving the model's performance is the assessment of alternative feature selection tools. The current classification model assesses the presence of information based on the available words in a piece of text. The selection of these words is of great importance to remove words with relatively little information from the text. For example, it is not desirable that a word like ”it” is assessed as an informative word, because it is clear that this is not a keyword. The current model still holds many non-informative words. To improve this, an extension of the selection methods is required. The current model only contains a stop word list and a stemming algorithm. The stop word list is a list of common words that have no value. The stemming algorithm ensures that equivalent words are conjugated to the stem of the word.

A first extension is a synonym list. Words like ”failure” and ”fault” can be treated equally. In addition, a semi-automatic selection of words is necessary. This can be supported by a dictionary containing subject-specific terms. Such terms are keywords such as abbreviations, system names and other words that are often used within the company. Furthermore, this method requires manual filtering of words from the final list to reduce the total number of keywords. A small selection of keywords with a lot of information that can make a clear distinction in the classification groups results in an accurate model. Aggarwal and Zhai (2012) have written a book in which they discuss many feature selection methods. This is a good starting point for improvement.

6.1.2. Reassessment of training and validation sets

The main improvement is the reassessment of the training and validation sets. This reassessment should be done for three different aspects: (1) the quality and detail of the problem description, (2) the size of the data set, and (3) the distinction of the class labels.

We recommend to start by improving the quality and detail of the problem descriptions in the current data set. The data holds many similar problem descriptions and many descriptions that have little information about the problem. A description like 'system is down' holds no information on the type of failure. The more information the data holds, the more information will be used to build the classification model. This increases the chance that the model will be able to properly classify new instances. For example, the chance that a new instance will hold a single word that is in the sentence ”system is down” is lower than the chance when looking at the sentence ”system is down, lights are flashing, system cannot be monitored”. We recommend to start improving the problem descriptions of three system types in the current data set and run the new data sets through the model. Possible improvements are the addition of words to the problem description to increase clarification, the exclusion of non-informative words, and the repair of grammatical errors. If the accuracy scores are higher, the hypothesis that more detail and better quality of the problem description influences the accuracy score is true. In that case, improve the other system types as well.

The new, more detailed data set will have to be reclassified, but first the class labels should be reassessed. This is a job for one or more experts, preferably a combination of Field Engineers and hardware/software developers. From the current selection of labels (*electrical failure, software failure, mechanical failure, human error, external damage,* and *no problem*), especially the labels *human error* and *no problem* were almost never correctly classified. To address the validity of the labels, the experts should answer questions like whether the current class labels are valid, whether all labels have a high distinction, and whether any more detail can be added to the class labels. After choosing the labels, each label should be tested on its accuracy score to show whether the label can be classified correctly.

The next step is to increase the data size. Even though the sensitivity analysis in section 5.3 did not show a significant difference in the accuracy scores of the data sets with different sizes,

it is evident that a larger data set holds a larger number of words and therefore has more information to build a model. The absolute increase of the accuracy score that results from a larger data set will differ per system type. Therefore, we recommend again to start with three system types and expand the data with an additional 50 instances. Then run the new data sets and test what the new accuracy scores are. If an improvement is observed, increase the size of the data set again with 50 instances. Repeat this until no significant increase of the accuracy score is observed. When the first increase in size of the data set already does not result in an increase of the accuracy score, the influence of the size of a data set is redundant for our type of data. When the best size of the data sets is determined, increase the size of the data sets for all system types. Now, the full data set is ready for the build of the classification model.

6.1.3. Improvement of the stemming algorithm

A more difficult, but also effective change is the improvement of the stemming algorithm. As mentioned in section 5.4, the current stemming algorithm *Snowball* has a high error rate for the Dutch language. Other stemming algorithms should be assessed. Snowball is the only Dutch stemming algorithm incorporated in the Knime Analytics software, so a new algorithm should be scripted and linked to the model. Knime Analytics offers the option to integrate such scripts from the languages *R* and *Python*. When a new algorithm is implemented, it should be tested on the accuracy score that results from the model.

Possible alternative stemming algorithms that support the Dutch language are analysed by Jonker (2019). The *Snowball*, *Porter*, and *Lancaster* algorithms are compared to a newly introduced algorithm called *Bag & Tag 'em*. A comparison is made based on understemming and overstemming (Table 6.1). Understemming is when the algorithm is not aggressive enough (i.e. words that should be stemmed are not stemmed), overstemming is when the algorithm is too aggressive (i.e. words that should not be stemmed, are stemmed). Table 6.1 shows that the newly introduced *Bag & Tag 'em* algorithm works best for understemming and the *Porter* algorithm works best for overstemming. On average, the *Bag & Tag 'em* algorithm has the highest accuracy score.

Table 6.1: Comparison of stemming algorithms

Algorithm	**Accuracy on understemming**	**Accuracy on overstemming**
Porter	90.66%	97.53%
Lancaster	92.62%	94.82%
Snowball	92.76%	96.98%
Bag & Tag 'em	96.78%	96.90%

6.1.4. Integration with the current ERP system

After improving the classification model, it must be integrated with the current ERP system. The main goal of this integration is to ensure the model requires a minimal number of user intermissions. In addition, it is required that the model runs without errors and has a fast feedback loop. The main tool used to build the classification model is Knime Analytics. This tool currently requires several intermissions by the user. To run a new data entry, this entry must be saved in a CSV file with the parameters *Service Order Number*, *System Type* and *Problem Text*. Then, the CSV file must be selected in Knime Analytics, workflow B (Figures 4.3 and 4.4). When the correct file is selected, the flow can be started. The output of the model is a CSV file with the parameters *Service Order Number*, *System Type*, *Problem Text*, *Probability class label A*, ..., *Probability class label F* and *Assigned Class*. As the free version of Knime Analytics does not support the use of a dashboard or report, we created

a (temporary) dashboard in Qlik Sense (Figure 6.2) to show what the final front-end of an implemented tool could look like.

This dashboard links the output CSV file from Knime Analytics to a full extract from the ERP system Navision. This full extract is the base file used in section 2.3 to analyse the performance parameters of the company. The dashboard provides an overview of the classified data linked to the trends of similar historic data based on the presence of four parameters: *System Group*, *System Type*, *Location Name*, and *Object Number*. For example, in Figure 6.2 the location name *V90546* is selected. The dashboard shows all newly classified service orders, the partial match data, and a set of historic trends belonging to the location name. At the top, we can also observe that this location holds an intersection system EC-2, which is maintained since 2012, only holds electrical failures and holds one object. A full overview of the dashboard is available in appendix B.

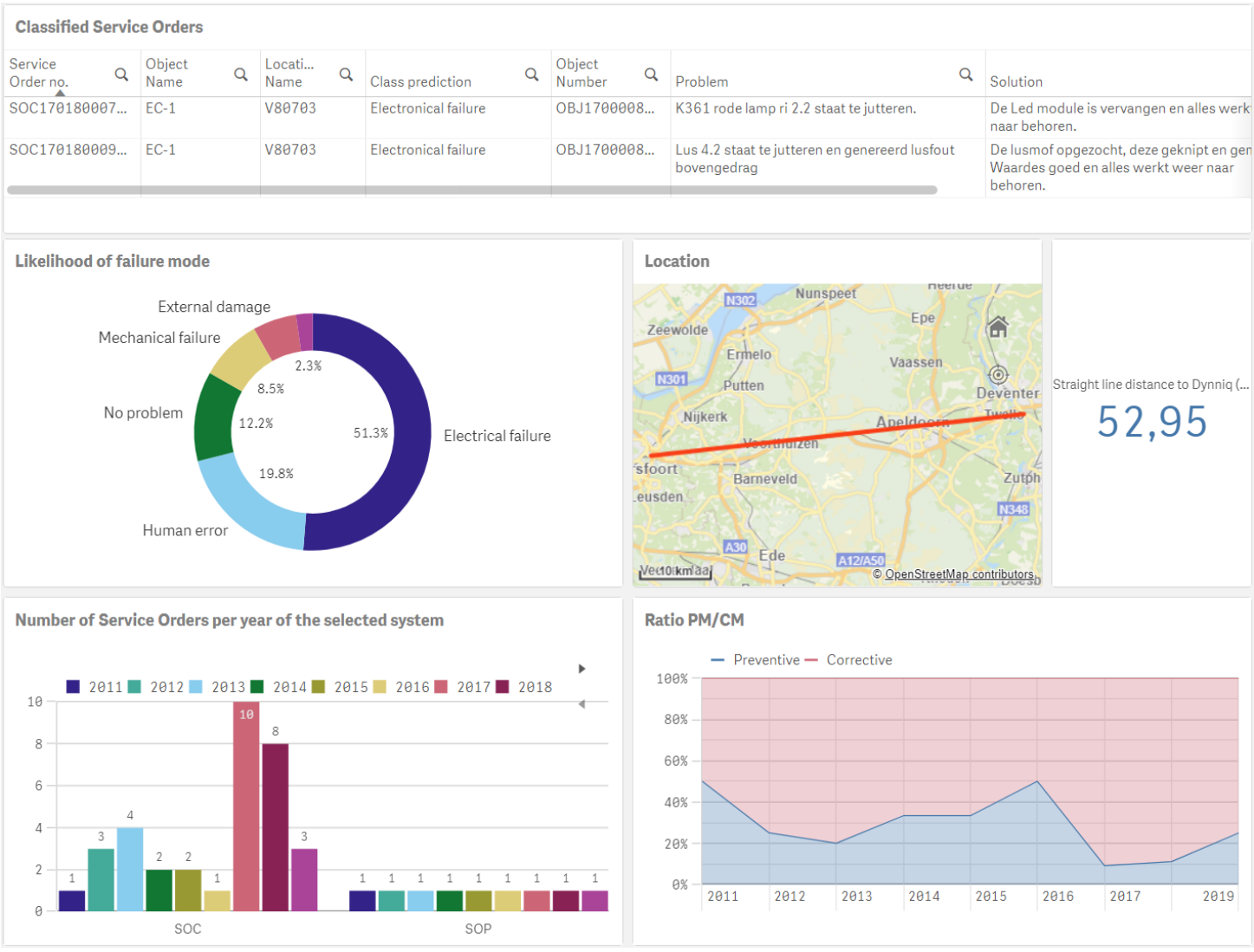


Figure 6.2: Truncated overview of dashboard for newly classified data

The number of user intermissions can be significantly reduced with a live connection between the database from Navision, the classification model in Knime Analytics, and the dashboard in Qlik Sense. Both Knime Analytics and Qlik Sense support live database connections. To ensure the model runs without error, a testing and optimization phase are recommended before and after implementation, respectively. It is also recommended to investigate the possibilities of creating a dashboard in the ERP system rather than in Qlik Sense. This will reduce the number of interfaces and speed up the feedback loop.

6.1.5. TIMELINE

A general timeline is created to give insight in the length of the optimization and integration phases. Incorporated in the timeline are the main steps discussed in this chapter and several go/no go moments in which the progress and feasibility of the implementation plan should be reviewed. This timeline should be reassessed before the execution of each step.

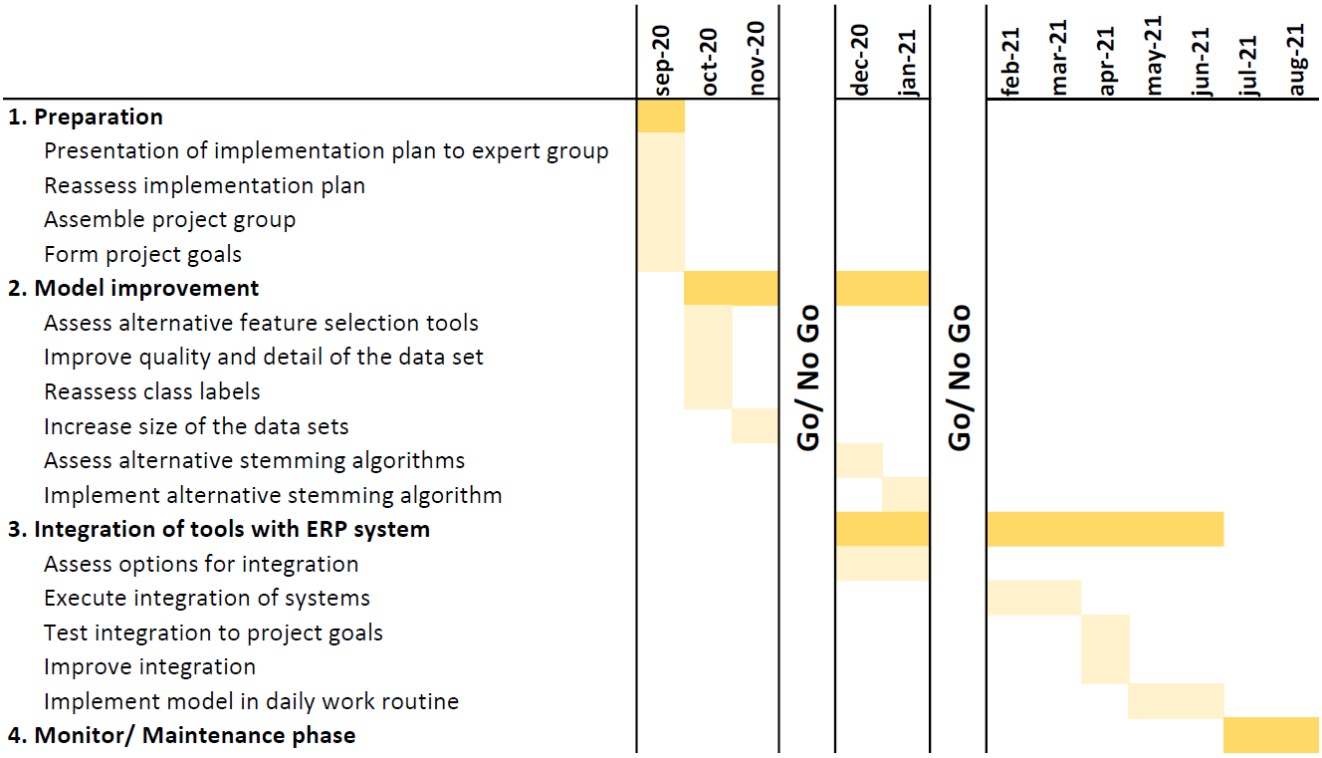


Figure 6.3: Timeline for implementation of the current model

6.2. RECOMMENDATIONS FOR FURTHER DEVELOPMENT

In section 6.1, we discussed changes to the model and the integration with the ERP system Navision. These steps are the required to implement an accurate model in the daily workload. However, this model can be further expanded in terms of user-friendliness and performance. In addition, new functionalities can be added by combining the text data with the structured and quantitative data from chapter 2. In this section we will discuss the recommendations for a future model. We start with a description of how the future model should be used. Then, we will discuss the possible adjustments and extensions.

6.2.1. OVERVIEW FUTURE MODEL

A Service Desk employee takes a new Service Order by telephone. The customer describes the fault he has observed and the Service Desk employee enters the problem description in the ERP system. The classification model provides live feedback on possible failure modes. In addition, all kinds of desired insights in the historical data are displayed for the system type to which the service order belongs (Figure 6.2). The Service Desk employee determines, based on the feedback of the dashboard, whether sufficient information is available to plan maintenance activities. This is not the case and therefore the problem is further analysed by asking the customer additional questions about possible failure modes. The customer has noticed a few things when monitoring the system remotely. The Service Desk employee expands the problem description and the classification model indicates a failure mode with sufficient certainty. The Service Desk employee reports directly to the customer that a problem solving approach will be started and plans a maintenance activity for a Field Engineer. The Field Engineer performs the maintenance activity based on the problem solving approach.

6.2.2. Research on alternative classification algorithms

As described in chapter 4, three classification algorithms were compared in the research: a Decision Tree (DT), a Support Vector Machine (SVM) and an Artificial Neural Network (ANN). The ANN was observed to be the most accurate algorithm. However, a very simple ANN has been used, which offers a large improvement potential for researching alternative machine learning algorithms. Faster algorithms as well as more accurate algorithms are desired. The book of Aggarwal and Zhai (2012) discusses several algorithms that can be used. This is a good starting point for the research.

6.2.3. Research on combining structured, and unstructured data

The current model uses the problem description as input data. This is just one of the many parameters entered during the registration of a Service Order. It is therefore interesting to investigate which information the other parameters contain and which information contains added value for the classification process. For example, if it is already known from a previous Service Order that a part may fail soon, it may be interesting to weigh it more heavily in the classification. It would also be an improvement to be able to predict not only the failure mode, but also which component has most likely failed.

6.2.4. Rewriting the classification model in OOP code

After implementation, the current model consists of three separate software packages: the ERP system (Navision), the text analysis tool (Knime Analytics), and the data visualisation tool (Qlik Sense). This makes the system prone to errors, for example due to updates in one of the software packages. The use of multiple tools also does not improve the speed of the feedback loop of the classification model. It is therefore recommended to rewrite the model in one of the Object Oriented Programming (OOP) languages (e.g. Python, Java, or Pascal). These languages are fast, easy to link to ERP systems and databases, and the model can be fully programmed as desired. In addition, OOP languages can be easily controlled from ERP systems.

7. Conclusions & Recommendations

In this final chapter, we will discuss the conclusions that reflect on the research questions from chapter 1. We will also discuss the limitations of the research and the designed model and we will end this chapter with recommendations for further development and implementation of the designed model.

7.1. Conclusions

This research is executed to solve the core problem at Dynniq Customer Service Operations: The maintenance strategy is expected not to be future proof. Underlying problems are the lack of knowledge on why systems fail and what information is in the maintenance logs that are created during maintenance activities. To solve this problem, we must answer the main research question:

"How can historical data on maintenance requests and solutions improve the effectiveness of any kind of maintenance, making it more future-proof?"

To answer this question, we analysed the performance of the company, found an improvement potential, executed a literature review to find methods for using the improvement potential, and designed and built a classification model that provides us with a problem solving approach. Based on these steps, we can draw the following conclusions that give answer to the main research question.

- The performance analysis showed that the extraction of information from unstructured text fields from the data set of maintenance requests and solutions, such as *problem description, cause,* and *solution*, can improve the lack of knowledge on why systems fail, why some of those failure cannot be solved at first attempt, and why preventive maintenance does not reduce the number of failures per system.
- The available data set of maintenance requests and solutions holds enough information on why systems fail in the unstructured text fields to classify Service Orders based on possible failure modes.
- The Artificial Neural Network (ANN) and Support Vector Machine (SVM) are the most accurate classifiers for the classification of the available unstructured text data based on types of failure modes.
- The ANN has the highest improvement potential as the current ANN algorithm can be extended with multiple hidden layers or replaced by a Recurrent Neural Network (RNN) which has the ability to repeat or improve calculations in the hidden layers within the same iteration.
- The failure modes from the ten different system types are too distinctive to combine the system types for the build of a single classification model. A separate classification model for each of the ten system types is therefore most accurate.
- Performance of the designed models is with 61% (ANN), 61% (SVM), and 55% (DT) similar to the reference articles. The ANN and SVM models perform slightly better than the more complex Recurrent Neural Network (RNN) to which the performance of the designed models is compared.

- Performance of the designed models is not high enough for direct implementation in the daily work routine. Improvements are required for feature selection, quality and detail of the data set, size of the data set, distinction of the class labels, and the stemming algorithm.

7.2. Limitations

During the research, several assumptions and concessions were made because of restrictions on time, available software, and available data. We will first discuss the most important limitations to the research and then the most important limitations to the designed classification model.

7.2.1. Limitations to the research

- The research has shown an improvement potential for both preventive and corrective maintenance, but the approach for designing a classification model only covers problem descriptions, which are often not present for preventive maintenance activities. This research therefore only solves a part of the problem cluster that is within the scope of this research.
- The literature review only includes classification algorithms that are available in free software packages while potentially more accurate machine learning algorithms are available for the classification of unstructured text data.
- The chosen class labels are based on a theory with high level failure modes, which do not provide detailed information on what part of the system has failed.

7.2.2. Limitations to the designed classification model

- The model uses feature selection tools which are not aggressive enough and therefore result in a set of features that still holds many uninformative terms.
- The model is trained and validated with a data set that holds a relatively low number of instances (500) for text classification algorithms.
- The test and validation set for training and validating the model is checked and classified by the researcher, who has no experience with executing the maintenance activities.
- The model uses a stemming algorithm that has a relatively high error rate for the Dutch language and does not incorporate Part-Of-Speech (POS) tagging.
- The model only uses single word features, which have a lower distinctive value than double or triple word features.
- The model is built with multiple software packages, which makes it prone to errors when used in the daily work routine and requires many user intermissions to fully run the classification model.

7.3. Recommendations

Given the conclusions and limitations, we can discuss several recommendations for further research and improvement of the designed model. We will discuss the most important recommendations.

- To ensure that the designed classification model provides enough information for improvement of the maintenance strategy, the model should not be implemented directly, but first requires several improvements to increase performance and user-friendliness. The following improvements to the model are recommended.

  - An assessment of alternative feature selection methods should be executed to decrease the number of uninformative features, which will result in an increase of the speed and accuracy of the designed classification model. Examples of unexplored feature selection methods are manual selection, synonym lists, and mathematical selection methods for quantifying the discrimination level of a feature through common methods like *Gini Index*, *Information gain*, or the χ^2*-statistic*.
  - A reassessment of the training and validation set, which is used for training and validating the classification model, should be executed to improve the data set on the aspects *quality and detail of the problem descriptions*, *size of the data set*, and *distinction of the class labels*.
  - An assessment of alternative stemming algorithms for the Dutch language is recommended to increase the percentage of correctly stemmed words.

- Alternative machine learning algorithms like Recurrent Neural Networks (RNNs) are available for the classification of unstructured text data and it is therefore recommended to extend the literature review to those algorithms and assess their performance potential in the designed classification model.
- Quantitative and structured data is available in the data set on maintenance requests and solutions. These data should be used to combine quantitative and structured parameters with the unstructured problem description for a more accurate or more targeted classification method.
- The build of a scripted version of the current classification model is recommended to increase the speed of the algorithm and improve the flexibility of the model for changes within the (classification and stemming) algorithms and the integration with the ERP system.

REFERENCES

Aggarwal, C. C. and Zhai, C. (2012). *Mining Text Data*. Springer, Dordrecht, Netherlands.

Beel, J., Gipp, B., Langer, S., and Breitinger, C. (2016). Research-paper recommender systems: a literature survey. *International Journal on Digital Libraries*, 17(4):305–338.

Edwards, B., Zatorsky, M., and Nayak, R. (2008). Clustering and classification of maintenance logs using text data mining. *Australasian Data Mining Conference*, 87:193–199.

Eikhart (2011). Dutch stopwords list. Retrieved from https://eikhart.com/blog/dutch-stopwords-list.

Jonker, A. (2019). Bag & tag 'em, a new dutch stemming algorithm. Retrieved from https://beta.vu.nl/nl/Images/stageverslag-jonker-anne.

Kantardzic, M. (2011). *Data Mining: Concepts, Models, Methods, and Algorithms*. John Wiley & Sons, Inc., Hoboken, NJ, United States, second edition.

Keerthi, S. S., Shevade, S. K., Bhattacharyya, C., and Murthy, K. R. K. (2001). Improvements to platt's smo algorithm for svm classifier design. *Neural Computation*, 13(3):637–649.

Larose, D. T. (2005). *Discovering knowledge in data, an introduction to data mining*. John Wiley & Sons, Inc., Hoboken, NJ, United States.

Law, A. M. (2015). *Simulation Modeling and Analysis*. McGraw-Hill Education, Tucson, AZ, United States, international edition.

Piatetsky, G. (2014). Crisp-dm, still the top methodology for analytics, data mining, or data science projects. Retrieved from https://www.kdnuggets.com/2014/10/crisp-dm-top-methodology-analytics-data-mining-data-science-projects.html.

Platt, J. C. (1999). Fast training of support vector machines using sequential minimal optimization. *Advances in Kernel Methods - Support Vector Learning*, pages 185–208.

Porter, M. (2002). Dutch stemming algorithm. Retrieved from http://snowball.tartarus.org/algorithms/dutch/ stemmer.html.

Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. Morgan Kaufmann Publishers, Inc., San Fransisco, CA, United States.

Quinlan, J. R. and Rivest, R. L. (1989). Inferring decision trees using the minimum description length principle. *Information and Computation*, 80:227–248.

Raileanu, L. E. and Stoffel, K. (2004). Theoretical comparison between the gini index and information gain criteria. *Annals of Mathematics and Artificial Intelligence*, 41(1):77–93.

Reshmy, A. K. and Paulraj, D. (2015). An efficient unstructured big data analysis method for enhancing performance using machine learning algorithm. *In Proceedings of the International Conference on Circuit, Power and Computing Technologies*, pages 1–7.

Riedmiller, M. and Braun, H. (1993). A direct adaptive method for faster backpropagation learning: the rprop algorithm. *IEEE International Conference on Neural Networks*, 1:586–591.

Rogalewicz, M. and Sika, R. (2016). Methodologies of knowledge discovery from data and data mining methods in mechanical engineering. *Management and Production Engineering Review*, 7(4):97–108.

Rumelhart, D. E. and McClelland, J. L. (1986). *Parallel distributed processing: explorations in the microstructure of cognition*. The MIT Press, Cambridge, MA, United States.

Salton, G. and McGill, M. J. (1983). *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, United States.

Salton, G., Wong, A., and Yang, C. S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620.

Sen, C., Hartvigsen, T., Kong, X., and Rundensteiner, E. (2019). Patient-level classification on clinical note sequence guided by attributed hierarchical attention. *International Conference on Big Data*, pages 930–939.

Shearer, C. (2000). The crisp-dm model: The new blueprint for data mining. *Journal for Data Warehousing*, 5(4):13–22.

Tan, Y., Zhang, C., Mao, Y., and Qian, G. (2015). Semantic presentation and fusion framework of unstructured data in smart cites. *In Proceedings of the 10th IEEE Conference on Industrial Electronics and Applications*, 10:897–901.

Tinga, T. (2013). *Principles of Loads and Failure Mechanisms*. Springer, Dordrecht, Netherlands.

van Bon, J. (2011). *ITIL A Pocket Guide - 2011 Edition*. Van Haren Publishing, Zaltbommel, Netherlands, first edition.

van Veen, F. (2017). Neural network zoo prequel: Cells and layers. Retried from https://www.asimovinstitute.org/author/fjodorvanveen/.

Wei, D., Wang, B., Lin, G., Liu, D., Dong, Z., Liu, H., and Liu, Y. (2017). Research on unstructured text data mining and fault classification based on rnn-lstm with malfunction inspection report. *Energies*, 10(3):406.

Appendices

A. KERNEL FUNCTION - NUMERICAL EXAMPLE

This appendix is an numerical example of Equations 3.8 to 3.12, which we extend to $n = 3$ and $m = 9$ instead of $n = 2$ and $m = 4$. We start with the general formula for a kernel function:

$$K(x, y) = \phi(x) \bullet \phi(y) = \phi(x)^T \phi(y) \tag{A.1}$$

where $K(x, y)$ is the kernel and x and y are n-dimensional inputs. ϕ is a $n \times m$ dimensional space, with m being the required dimension for separation. Normally a function like $\phi(x)^T \phi(y)$ requires us to first calculate $\phi(x)$ and $\phi(y)$ and then calculate the dot product of the functions. This can be simplified by using a corresponding kernel. For example, we have the following two vectors:

$$x = (x_1, x_2, x_3)^T = (1, 2, 3)^T \tag{A.2}$$

$$y = (y_1, y_2, y_3)^T = (4, 5, 6)^T \tag{A.3}$$

Lets say a linear separator is possible in the ninth dimension (i.e. $m = 9$). Then, $\phi(x)$ and $\phi(y)$ become:

$$\phi(x) = (x_1^2, x_1x_2, x_1x_3, x_2x_1, x_2^2, x_2x_3, x_3x_1, x_3x_2, x_3^2) \tag{A.4}$$

$$\phi(y) = (y_1^2, y_1y_2, y_1y_3, y_2y_1, y_2^2, y_2y_3, y_3y_1, y_3y_2, y_3^2) \tag{A.5}$$

Then, we plug in the numbers for x and y.

$$\phi(x) = (1, 2, 3, 2, 4, 6, 3, 6, 9) \tag{A.6}$$

$$\phi(y) = (16, 20, 24, 20, 25, 30, 24, 30, 36) \tag{A.7}$$

and Equation A.1 becomes:

$$K(x, y) = 16 + 40 + 72 + 40 + 100 + 180 + 72 + 180 + 324 = 1024 \tag{A.8}$$

This equation can be simplified to a two-dimensional form, the kernel, which reaches the same results:

$$K(x, y) = (x^T y)^2 = (4 + 10 + 18)^2 = 32^2 = 1024 \tag{A.9}$$

B. DASHBOARD FOR NEWLY CLASSIFIED DATA

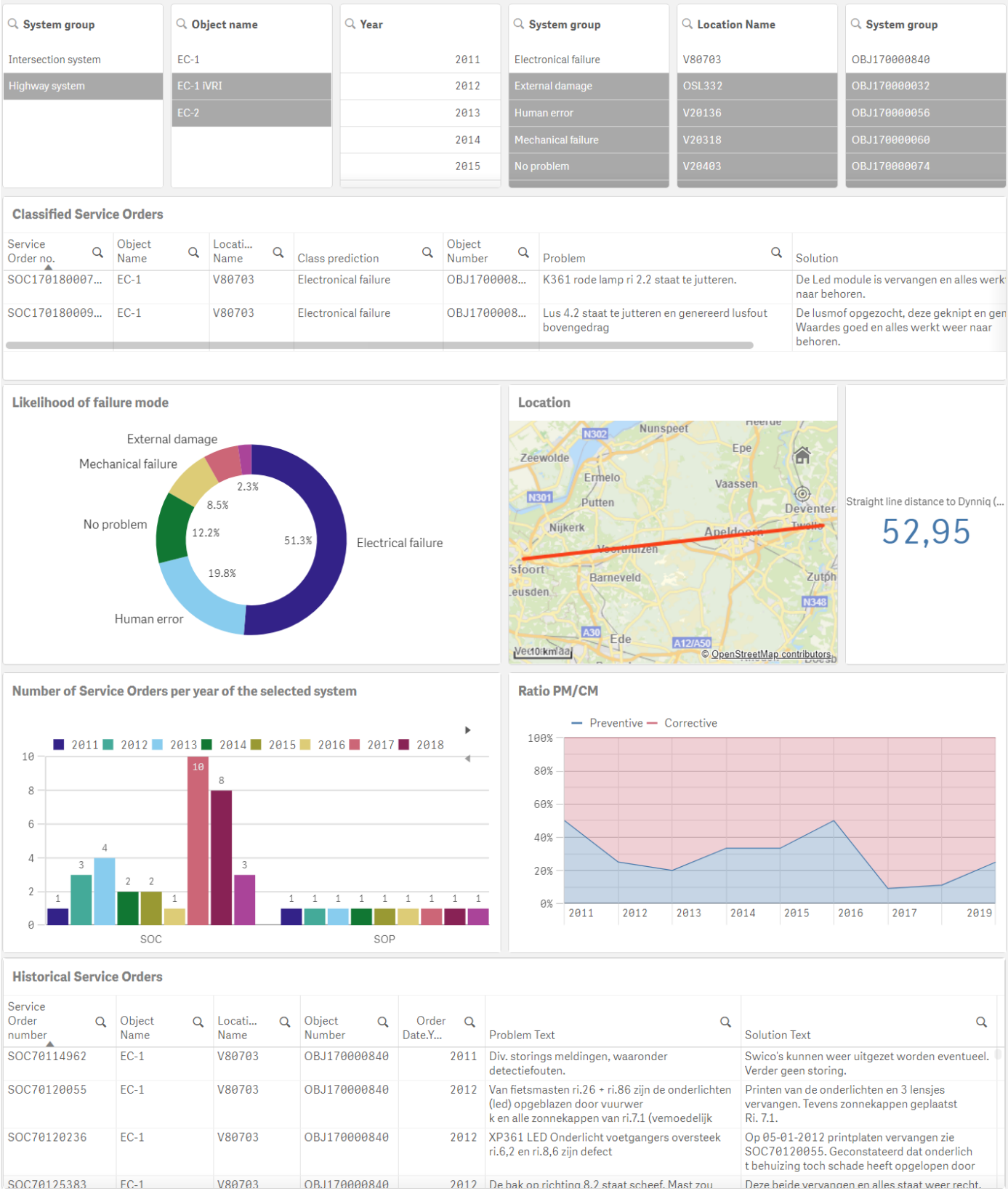


Figure B.1: Overview dashboard for newly classified data

C. STOP WORD LIST

a
aan
aangaande
aangezien
achter
achterna
aen
af
afd
afgelopen
agter
al
aldaar
aldus
alhoewel
alias
alle
allebei
alleen
alleenlyk
allen
alles
als
alsnog
altijd
altoos
altyd
ander
andere
anderen
anders
anderszins
anm
b
behalve
behoudens
beide
beiden
ben
beneden
bent
bepaald
beter
betere
betreffende
bij
bijna
bijvoorbeeld
bijv
binnen
binnenin
bizonder
bizondere
bl
blz
boven
bovenal
bovendien
bovengenoemd
bovenstaand
bovenvermeld
buiten
by
daar
daarheen
daarin
daarna
daarnet
daarom
daarop
daarvanlangs
daer
dan
dat
de
deeze
den
der
ders
derzelver
des
deszelfs
deszelvs
deze
dezelfde
dezelve
dezelven
dezen
dezer
dezulke
die
dien
dikwijls
dikwyls
dit
dl
doch
doen
doet
dog
door
doorgaand
doorgaans
dr
dra
ds
dus
echter
ed
een
eene
eenen
eener
eenig
eenige
eens
eer
eerdat
eerder
eerlang
eerst
eerste
eersten
effe
egter
eigen
eigene
elk
elkanderen
elkanderens
elke
en
enig
enige
enigerlei
enigszins
enkel
enkele
enz
er
erdoor
et
etc
even
eveneens
evenwel
ff
gauw
ge
gebragt
gedurende
geen
geene
geenen
gegeven
gehad
geheel
geheele
gekund
geleden
gelijk
gelyk
gemoeten
gemogen
geven
geweest
gewoon
gewoonweg
geworden
gezegt
gij
gt
gy
haar
had
hadden
hadt
haer
haere
haeren
haerer
hans
hare
heb
hebben
hebt
heeft
hele
hem
hen
het
hier
hierbeneden
hierboven
hierin
hij
hoe
hoewel
hun
hunne
hunner
hy
ibid
idd
ieder
iemand
iet
iets
ii
iig
ik
ikke
ikzelf
in
indien
inmiddels
inz
inzake
is
ja
je
jezelf
jij
jijzelf
jou
jouw
jouwe
juist
jullie
kan
klaar
kon
konden
krachtens
kunnen
kunt
laetste
lang
later
liet
liever
like
m
maar
maeken
maer
mag
martin
me
mede
meer
meesten
men
menigwerf
met
mezelf
mij
mijn
mijnent
mijner
mijzelf
min
minder
misschien
mocht
mochten
moest
moesten
moet
moeten
mogelijk
mogelyk
mogen
my
myn
myne
mynen
myner
myzelf
na
naar
nabij
nadat
naer
net
niet
niets
nimmer
nit
no
noch
nog
nogal
nooit

nr
nu
o
of
ofschoon
om
omdat
omhoog
omlaag
omstreeks
omtrent
omver
onder
ondertussen
ongeveer
ons
onszelf
onze
onzen
onzer
ooit
ook
oorspr
op
opdat
opnieuw
opzij
opzy
over
overeind
overigens
p
pas
pp
precies
pres
prof
publ
reeds
rond
rondom
rug
s
sedert
sinds
sindsdien
sl
slechts
sommige
spoedig
st
steeds
sy
t
tamelijk
tamelyk
te
tegen
tegens
ten
tenzij
ter
terwijl
terwyl
thans
tijdens
toch
toe
toen
toenmaals
toenmalig
tot
totdat
tusschen
tussen
tydens
u
uit
uitg
uitgezonderd
uw
uwe
uwen
uwer
vaak
vaakwat
vakgr
van
vanaf
vandaan
vanuit
vanwege
veel
veeleer
veelen
verder
verre
vert
vervolgens
vgl
vol
volgens
voor
vooraf
vooral
vooralsnog
voorbij
voorby
voordat
voordezen
voordien
voorheen
voorop
voort
voortgez
voorts
voortz
vooruit
vrij
vroeg
vry
waar
waarom
wanneer
want
waren
was
wat
we
weer
weg
wege
wegens
weinig
weinige
wel
weldra
welk
welke
welken
welker
werd
werden
werdt
wezen
wie
wiens
wier
wierd
wierden
wij
wijzelf
wil
wilde
worden
wordt
wy
wyze
wyzelf
zal
ze
zeer
zei
zeker
zekere
zelf
zelfde
zelfs
zelve
zelven
zelvs
zich
zichzelf
zichzelve
zichzelven
zie
zig
zij
zijn
zijnde
zijne
zijner
zo
zo’n
zoals
zodra
zommige
zommigen
zonder
zoo
zou
zoude
zouden
zoveel
zowat
zulk
zulke
zulks
zullen
zult
zy
zyn
zynde
zyne
zynen
zyner
zyns

D. SNOWBALL STEMMING ALGORITHM

```
routines (
           prelude postlude
           e_ending
           en_ending
           mark_regions
           R1 R2
           undouble
           standard_suffix
)

externals ( stem )

booleans ( e_found )

integers ( p1 p2 )

groupings ( v v_I v_j )

stringescapes {}

/* special characters (in ISO Latin I) */

stringdef a"   hex 'E4'
stringdef e"   hex 'EB'
stringdef i"   hex 'EF'
stringdef o"   hex 'F6'
stringdef u"   hex 'FC'

stringdef a'   hex 'E1'
stringdef e'   hex 'E9'
stringdef i'   hex 'ED'
stringdef o'   hex 'F3'
stringdef u'   hex 'FA'

stringdef e`   hex 'E8'

define v       'aeiouy{e`}'
define v_I     v + 'I'
define v_j     v + 'j'

define prelude as (
    test repeat (
        [substring] among(
            '{a"}' '{a'}'
                (<- 'a')
            '{e"}' '{e'}'
                (<- 'e')
            '{i"}' '{i'}'
                (<- 'i')
            '{o"}' '{o'}'
                (<- 'o')
            '{u"}' '{u'}'
                (<- 'u')
            ''  (next)
        ) //or next
    )
    try(['y'] <- 'Y')
    repeat goto (
        v ['i'] v <- 'I') or
          ('y']   <- 'Y')
    )
)

define mark_regions as (

    $p1 = limit
    $p2 = limit

    gopast v  gopast non-v  setmark p1
    try($p1 < 3  $p1 = 3)  // at least 3
    gopast v  gopast non-v  setmark p2

)

define postlude as repeat (

    [substring] among(
        'Y'  (<- 'y')
        'I'  (<- 'i')
        ''   (next)
    ) //or next
)

backwardmode (

    define R1 as $p1 <= cursor
    define R2 as $p2 <= cursor

    define undouble as (
        test among('kk' 'dd' 'tt') [next] delete
    )

    define e_ending as (
        unset e_found
        ['e'] R1 test non-v delete
        set e_found
        undouble
    )

    define en_ending as (
        R1 non-v and not 'gem' delete
        undouble
    )

    define standard_suffix as (
        do (
            [substring] among(
                'heden'
                (   R1 <- 'heid'
                )
                'en' 'ene'
                (   en_ending
                )
                's' 'se'
                (   R1 non-v_j delete
                )
            )
        )
        do e_ending

        do ( ['heid'] R2 not 'c' delete
             ['en'] en_ending
           )

        do (
            [substring] among(
                'end' 'ing'
                (   R2 delete
                    (['ig'] R2 not 'e' delete) or undouble
                )
                'ig'
                (   R2 not 'e' delete
                )
                'lijk'
                (   R2 delete e_ending
                )
                'baar'
                (   R2 delete
                )
                'bar'
                (   R2 e_found delete
                )
            )
        )
        do (
            non-v_I
            test (
                among ('aa' 'ee' 'oo' 'uu')
                non-v
            )
            [next] delete
        )
    )
)

define stem as (

        do prelude
        do mark_regions
        backwards
            do standard_suffix
        do postlude
)
```

E. Confusion matrices Class Labels

Table E.1: Confusion matrix of actual (left) and predicted (top) failure modes for ANN model

	Electrical failure	**Software failure**	**Mechanical failure**	**External damage**	**Human error**	**No problem**
Electrical failure	73%	18%	1%	2%	2%	4%
Software failure	23%	66%	1%	2%	3%	5%
Mechanical failure	26%	16%	36%	5%	4%	13%
External damage	24%	11%	2%	52%	1%	11%
Human error	40%	31%	2%	1%	24%	2%
No problem	41%	30%	3%	10%	0%	15%

Table E.2: Confusion matrix of actual (left) and predicted (top) failure modes for SVM model

	Electrical failure	**Software failure**	**Mechanical failure**	**External damage**	**Human error**	**No problem**
Electrical failure	66%	33%	0%	1%	0%	0%
Software failure	28%	70%	0%	1%	0%	0%
Mechanical failure	37%	55%	9%	0%	0%	0%
External damage	43%	37%	0%	17%	0%	3%
Human error	41%	44%	2%	0%	12%	1%
No problem	51%	48%	0%	2%	0%	0%

Table E.3: Confusion matrix of actual (left) and predicted (top) failure modes for DT model

	Electrical failure	**Software failure**	**Mechanical failure**	**External damage**	**Human error**	**No problem**
Electrical failure	77%	22%	0%	0%	0%	1%
Software failure	28%	70%	0%	0%	0%	1%
Mechanical failure	34%	33%	30%	3%	0%	1%
External damage	41%	26%	0%	32%	0%	1%
Human error	45%	47%	0%	0%	7%	0%
No problem	57%	34%	0%	9%	0%	0%