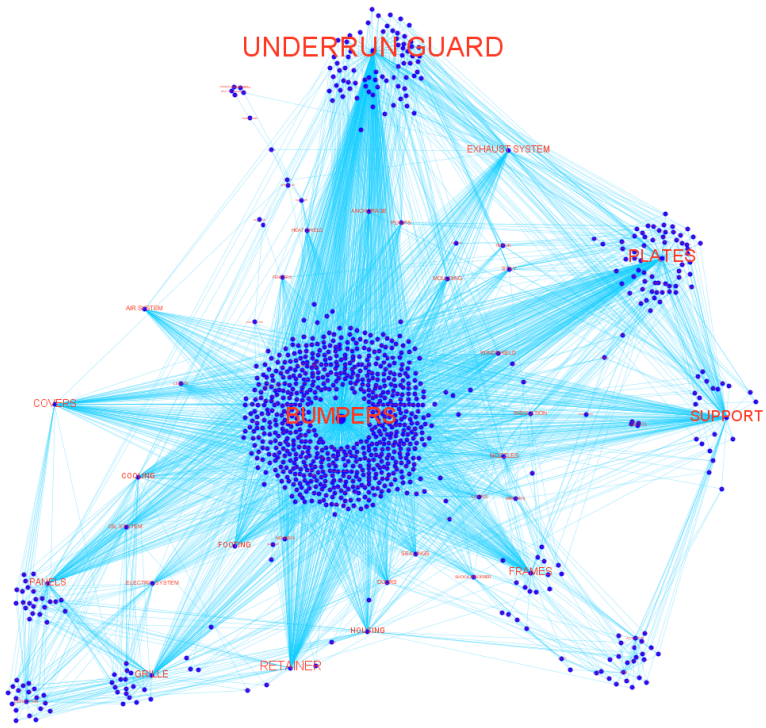
CHALMERS
UNIVERSITY OF TECHNOLOGY



On Data-Driven Predictive Maintenance of Heavy Vehicles

A case study on Swedish trucks

Master's thesis in Complex Adaptive Systems

ANDREAS FALKOVÉN

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2017

Master's thesis EX025/2017

On Data-Driven Predictive Maintenance of Heavy Vehicles

A case study on Swedish trucks

ANDREAS FALKOVÉN

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Signal Processing and Biomedical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2017

On Data-Driven Predictive Maintenance of Heavy Vehicles
A case study on Swedish trucks
ANDREAS FALKOVÉN

Supervisor: Ulf Lindgren, Combitech AB
Examiner: Lennart Svensson, Department of Electrical Engineering

Master's Thesis EX025/2017
Department of Electrical Engineering
Division of Signal Processing and Biomedical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Graphical representation of some of the association rules between part replacements made on trucks.

Typeset in LaTeX
Gothenburg, Sweden 2017

On Data-Driven Predctive Maintenance of Heavy Vehicles
A case study on Swedish trucks
ANDREAS FALKOVÉN
Department of Electrical Engineering
Chalmers University of Technology

Abstract

This thesis studies whether data collected from Volvo trucks in Sweden can be used for the purpose of predictive maintenance. The goal has been to see whether this data can be used to predict when service is required in a truck, rather than at set intervals or after a fault has occurred. To determine if the provided data can be used to predict and diagnose fault, three different approaches to analysing the data are tested. Association rule mining and classification trees are used to investigate if maintenance requirements can be detected based on relations between different maintenance types. Autoregressive modelling of parameters that are monitored by a truck is also attempted, in order to see whether these data series can be reliably predicted and forecasted. The results found using association rule learning and classification trees show that it is to some degree possible to detect fault requirements, although not to a reliable extent, using the provided data. Autoregressive modelling suffers from a lack of usable information in the data, and highlight some issues with using the provided data for predictive maintenance. The main conclusion from this thesis is that there is information relevant for implementing a predictive maintenance framework contained in the provided data. However, the overall quality and relevance of the provided data was not sufficient for a reliable implementation of such a framework as things stand. Before a predictive maintenance framework can be implemented, this thesis shows that a reevaluation of the underlying data is required.

Keywords: predictive maintenance, data mining, association rule learning, classification trees, autoregressive modelling, diagnostics

Acknowledgements

First and foremost I would like to express my greatest gratitude to my supervisor at Combitech, Ulf Lindgren, for his support during the thesis work. His inputs and discussions during the course of the work and thesis writing have been both insightful and inspiring, opening my eyes to new ways of viewing what I have learned at Chalmers. Secondly I would like to thank associate professor Lennart Svensson for helping to steer me in the right directions with his feedback during the project. Thirdly, I would like to extend a collective acknowledgement to those at AB Volvo whom I have been in contact with during the thesis work. Whether for enabling me to perform this thesis work, discussing outcomes during the work, or allowing me to pretend to be a truck mechanic for a day, a great thanks to you all. Last but not least I would like to extend an informal thank you to my friends at Chalmers for providing a necessary distraction from my work from time to time.

Andreas Falkovén, Gothenburg, May 2017

Contents

1

Introduction and Background

This introductory chapter provides a presentation of the background and goals of this thesis project. To start with, an introduction to different approaches to maintenance of technical systems, such as the trucks investigated later in the thesis, is presented. In particular, a more thorough review of predictive maintenance and different approaches to this paradigm is given. This also includes a review of previous works in the fields of predictive maintenance in particular. Finally, the main objectives and limitations of this project are stated, along with an outline of the rest of the thesis.

1.1 Maintenance Schemes for Industrial Systems

In many industrial systems, such as production lines and machinery, it is vital that the personnel operating or using these systems can do so safely and reliably. Moreover, from an economical and sustainability standpoint it is also essential that a system remains in operable condition for as long as possible. A reliable system prevents loss of productivity due to malfunctions, wear and tear or other forms of degradation that leaves the system being unable to fulfil its intended purpose. For example, an industrial bearing is subject to constant strain due to vibrations, friction and fatigue in the materials, which may eventually cause the bearing to fail. A failing bearing could in turn lead to an emergency stop of the operating machinery, resulting in a halt in production and significant costs in lost income. To address these issues, a variety of maintenance schemes are deployed once a system is in service. This allows suitable actions to be taken to ensure that it can remain usable for as long as possible. Such schemes can roughly be divided into three paradigms, based on why, when and how the maintenance is carried out; corrective- , planned- and predictive maintenance [1].

In *corrective* maintenance, service and repairs are carried out once a fault or malfunction has already occurred in a system, or when it is otherwise unable to fulfil its intended purpose. For example, an axle in a machine that is replaced only once it breaks, or a drill which is replaced once it has gone dull, would be subjected to corrective maintenance. When a fault has occurred, the malfunctioning part is identified and either restored to operable condition or replaced. Corrective maintenance

naturally minimises the number of unnecessary part replacements or repairs, since maintenance is only carried out as needed. However, this approach can also lead to unexpected and lengthy losses of production, safety risks as a system nears failure, or expensive repairs and replacements.

In contrast to corrective maintenance, *planned* maintenance schemes maximise system reliability by scheduling maintenance at specific intervals, before any malfunctions take place. Such intervals could be determined by, for example, a fixed number of operating hours, operating cycles or a certain mileage. For systems where reliable performance is critical, or when there are severe consequences of a failure, a preventive approach is generally preferred over a corrective one. However, the implementation of a good preventive maintenance scheme often requires a deep understanding of the system at hand, in order to be able to specify the maintenance intervals. Such knowledge could be based on lifetime estimations, historical performance measures or comparisons with similar systems. Due to the dynamic nature of an industrial system and the influence of many different environmental effects in its surroundings, finding suitable service intervals is often difficult. This can lead to very conservative measures being used, causing parts and components to be replaced despite being in functional condition. This in turn increases maintenance costs and requires large supplies of service materials to be readily available [2].

An additional aspect to when maintenance is carried out is when equipment failures are expected in a system. Figure 1.1 illustrates a number of different equipment failure models. Such models estimate the instantaneous probability of equipment failure over time. While a system may experience a fault at any time, there is typically periods during its lifetime where failures are more common. A successful maintenance scheme will thus take these periods into consideration when scheduling maintenance of a system. Referring to figure 1.1, line A represents a system where failure is equally probable during the entire lifetime of the system. In such a system, faults occur randomly. Lines B and C are instead the failure rate of a system where early life faults are probable (B), or where faults are increasingly probable with age (C) due to wear. Line D shows a system which behaves like a combination of the previous three, having a higher failure rate at the beginning and end of its lifetime, but having a constant, lower failure rate in between [3]. A planned maintenance scheme would typically aim at performing maintenance before the increasing failure probability at the end of system life becomes too high. However, determining this time is not trivial, and requires experience and empirical evidence of when faults occur.

1.1.1 The Predictive Maintenance Paradigm

As described in the previous section, both corrective and preventive maintenance approaches come with different pros and cons. By combining aspects from both these maintenance paradigms the *predictive*, also known as *condition based*, maintenance paradigm is obtained. This paradigm seeks to capture the positive aspects of both previously mentioned maintenance approaches, while at the same time eliminating

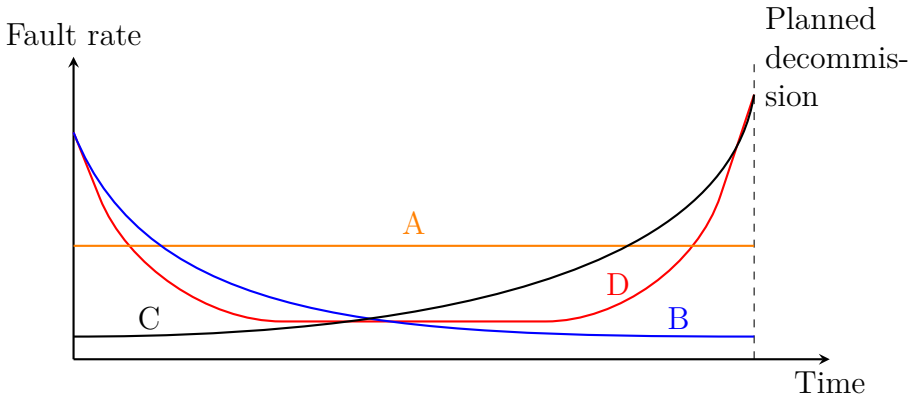


Figure 1.1: Different schematic models of the fault probability in some type of system during its lifetime, based on hazard models. Line A represents a system where faults occur randomly with constant probability during the entire life cycle of the system. Line B represents a system with a high initial failure rate, which then reduces over time as the system is "burned in". On the contrary, line C represents the opposite type of behaviour, where the product has a low risk of faults when new but where the probability of faults occurring increases up over time. Lastly, line D represent a combination of the previous three, where a system has an increased fault probability at the start and end of its life cycle, but a lower, constant failure rate in between.

the drawbacks. As the name implies the idea behind predictive maintenance is to predict when maintenance is required based on the actual condition of a system. Replacements, service or repairs are then carried out only when the system condition indicates that failure or malfunction is imminent. By doing this, the goal is to maximise both system availability and functionality. At the same time, unexpected failures associated with corrective maintenance and the redundant replacements associated with preventive maintenance are kept to a minimum [4]. The success of such predictive maintenance relies on a few premises. First, it must be possible to monitor a system in a way which allows *predicting* when a failure will occur. Second, it must also be possible to accurately *diagnose* such a failure once predicted. At last, one would also like to obtain decision support regarding what action to take, based on predictions and diagnosis made in the previous two steps.

Information about the system can be obtained in a variety of ways. It can consist of historical data such as repair logs and previous failure conditions. It can also be obtained directly from the system in the form of sensor signals, such as vibration data or electrical currents, and measurements such as acoustic noise, heat levels or optical observations. This data can in turn be read either in real time, *i.e.* on-line tracking, or come from data files collected over time, *i.e.* off-line tracking. An on-line implementation gives more up-to-date data, which is crucial in fast paced or time restrained operations, whereas an off-line implementation uses data which may be collected at any time during the equipment's life cycle. However, processing large amounts of data in real time requires larger amounts of computing power, a resource which may be scarce in for example smaller, embedded systems. In such cases it may

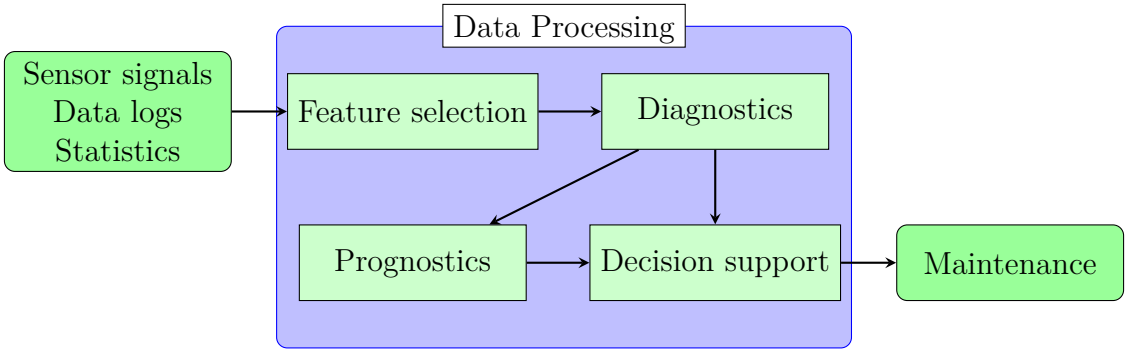


Figure 1.2: Flowchart of a typical predictive maintenance environment. Data is acquired from a system and fed into a program for feature selection for pre-processing. The processed data is then forwarded to a program which calculates and classifies the current health condition of the system. This information is in turn sent to two modules for prognostics and decision support. The process of prognostics is in turn meant to estimate the time remaining until maintenance is required, *i.e.* determining the system's remaining useful life (RUL). In the decision support module this estimation is used together with the diagnostics data to give a basis for deciding if maintenance is currently necessary, which in turn may result in some form of reactive action.

be preferable to use an off-line implementation which can process the data supplied in an outside environment.

Figure 1.2 shows an outline of how a typical predictive maintenance framework is organised. Such an environment can divided into of a number of stages, each of which is helping to model and interpret the current health condition of an asset [5]. The function of these stages is explained in detail in the subsequent sections.

1.1.2 Feature Selection and Data Fusion

In the *feature selection* step, various methods are used to identify which of the incoming signals from the equipment carry information about the system condition. This process is much akin to how significant variables are identified in statistical analysis. Typically, this step will involve filtering noise from data, performing sensor- and/or data fusion and then selecting suitable health condition indicators. The type of pre-processing that is required is data dependent, as different data sources carry with them varying levels of noise and precision. For example, in vibration data one might want to filter out high frequency modes from small vibrations, and in thermal imaging data one might want to compensate for ambient temperature. Data- and sensor fusion aims to combine information from different sources in order to aggregate the data, reduce the dimensionality of the data and/or to allow different sources to complement each other [6]. If raw data is collected directly from the system one might for example want to combine data from different subsystems to obtain a single aggregated signal describing the entire system.

1.1.3 Prognostic and Diagnostic Approaches

Diagnostics and *prognostics* are the workhorses of a predictive maintenance framework. Diagnostics in this context relates to identifying and classifying faults that occur in a system. The process of prognostics is in turn meant to estimate the time remaining until such faults actually occur, *i.e.* determining the system's remaining useful life (RUL) [7]. Depending on what methods are used for these purposes, diagnostics and prognostics can be divided into a number of sub-categories; they can be experience based, knowledge based, model based or data-driven, as outlined in [2] amongst others.

Experience based approaches are in general the simplest and, as the name implies, are based on experience in the form of performance and fault history of a piece of equipment [5]. These experience based models use such historical data to fit a statistical model of the failure-rate of the equipment. This model can then be used to estimate the probability of failure within a given time frame. Since these models use historical equipment data to fit a statistic model it is crucial that the data is representative of system behaviour if the model is to be reliable for prediction and diagnostics.

Model based approaches use mathematical models to describe the physical behaviour of a system. By modelling the expected behaviour of the system, one relies on the ability to detect faults and malfunctions by observing deviations from the expected behaviour of the system. If these deviations are large, as determined by statistical testing, one would expect the system to be behaving abnormally. Observing how deviations change over time makes it possible to estimate future changes in the system [8]. One large benefit of using a model based prediction approach is the ability to update and change the model. This allows incorporating new features of the system, or known changes in its behaviour and operating conditions, directly into the model. However, this also means that creating an accurate model of a system requires knowledge of the failure mechanics and how the specific system works. The accuracy of these models are therefore strongly dependent on the understanding of the monitored system [5].

Due to the difficulty in creating good models of many complex physical real life systems, *knowledge based* approaches forego this limitation by instead attempting to mimic human expertise about the system. *Expert systems* (ES) and *fuzzy logic* (FL) units are examples of methods belonging to this kind of prediction approach [5]. These methods use simple decision rules to determine the state of a system much like a human expert would. While ES methods such as classification and regression trees (CART) have historically been a popular method for performing predictions and diagnostics [2], they do have their drawbacks. An ES cannot make proper decisions in situations that have not been included in the knowledge base on which it has been trained, and obtaining such a knowledge base is a problem in itself. Moreover, increasing the number of decision rules used by the ES eventually leads to a large increase in computational power required due to the combinatorial explosion of possible decision paths [2]. Fuzzy logic can be seen as a generalisation

of ordinary Boolean logic, which uses several "levels" of certainty in addition to the conventional "true" and "false". Thus, in fuzzy logic one can model several diffuse states of a system, such as "low", "normal", and "high", rather than just two distinct states. The concept of fuzzy logic is often used in conjunction with some other method such as an expert system or Kalman filter in order to enable modelling states of degradation of a system [7].

Data driven methods revolve around processing data collected from a system, using either statistical or artificial intelligence techniques [2]. Commonly used statistical techniques include regression models, principal component analysis (PCA), vector quantisation (VQ), and hazard rate analysis. State space models such as hidden- and hidden semi-Markov models (HMM and HSMM, respectively) or dynamic Bayesian Networks are other commonly used statistical techniques. Artificial intelligence techniques generally involve variants of artificial neural networks (ANN), such as recurrent neural networks, dynamic wavelet networks or self organising maps [5]. Statistical models aim to capture trends in the supplied data and use these trends to estimate the current state of the system. Statistical models also seek to determine a remaining life distribution of the system. In *HMMs* and *HSMMs* the hidden states are used to model some underlying state of degradation in a system, and the current state is estimated using the observed data. By estimating state transition times these models are also able to give a measure of the RUL of the system [2]. While being considered powerful methods for modelling system degradation [8], the major drawback of HMMs is that the underlying mathematical models are based on assumptions which are rarely applicable in real-life situations. These assumptions include the observations being independent, as well as the Markov assumption that the probability of the system being in a state at some time T depends only on the state at time $T-1$ [9]. *Hazard rate models* are methods from survival analysis. These models use supplied data to fit two probability distributions, a lifetime distribution and a reliability distribution. In [2], environmental variables are also incorporated into a hazard rate model which is then used to obtain a probability distribution over the system's current state and RUL.

Artificial intelligence techniques are data driven methods which utilise computational structures based on different neural network architectures. The power behind these methods is twofold. With the right choice of architecture, ANNs are able to act as both non-linear classifiers as well as models of the dynamical time evolution of a system. Thus these models are able to both recognise and classify patterns in signals from a system, and at the same time predict future system behaviour [2]. The biggest strength of both statistical and AI techniques are that, with the correct training schemes, they are able to tune their own parameters. Thus they do not require any specific knowledge of the system at hand and are able to adapt to changes in system conditions. However, this ability comes with the drawback that these methods require large amounts of representative data which they can train on before being able to provide usable outputs. In addition, neural network models are prone to overfitting the training data without proper measures being taken, as described in [10].

The above approaches are not necessarily used on their own. Using combinations of different methods can lead to improved models, which are able to better capture the behaviour of a complex system. It is also possible to use one method for optimal diagnostic ability and another method for optimal predictive ability [11]. Further, a suitable choice of method is largely dependent on what data is available. With a lack of sensor data, a statistical approach based on historical records is a suitable approach. An ANN approach could instead be a better choice if large amounts of system data can be collected.

1.1.4 Decision Support and Maintenance Action

The last step in a predictive maintenance framework involves turning the results from the previous steps into a recommendation for what maintenance action to take. When a fault is predicted, a suitable response based on the time remaining until it occurs should be given. This enables a professional to perform the required maintenance on the system at an optimal time. If the problem is estimated to be far off into the future, an indication of when service will be required could be a good way of allowing spare parts to be ordered in time, or for other maintenance requirements to be prepared. Furthermore, having the same faults consistently identified could be an indicator that some form of system redesign is suitable.

1.2 Objective of This Thesis

The main objective of this thesis was to investigate whether diagnostic data collected by AB Volvo can be used to predict and diagnose failing components in trucks. This meant being able to either provide a reliable proof-of-concept that the collected data can indeed be used for predictive maintenance, or coming to the conclusion that the provided data cannot be used for this purpose. In the former case, a model or method with the possibility of being generalised to trucks in service was to be provided, while in the latter case arguments as to why the data is insufficient and suggestions for additional data requirements were sought.

To fit within the scope of a master's thesis, a few limitations have been made during the project. One such limitation is that the data was limited to only that collected from trucks operating in Sweden during 2016. This limit was made by the data supplier AB Volvo who believed that this would still give a large diversity of trucks with different operational conditions, while limiting the amount of data sources to a representable subset of the entire truck fleet. Additionally, the number of parameters used during analysis and modelling of the provided data was limited to a few manually selected ones. This was done to keep work focused, allowing more time to be used for analysing the results.

1.3 Previous Works

There is a large amount of literature available regarding implementations of predictive maintenance using different approaches and systems. This section describes some articles on the subject that were used as a basis for suitable modelling and methodology options in this thesis.

HMMs have been used by many researchers for trying to associate system signal patterns with underlying states of degradation. In [12] an ensemble of HMMs are used in conjunction with PCA and VQ to both diagnose and predict the future health state of an experimental setup. In [13] a technique involving competitively trained HMMs to characterise each state of degradation in drill-bits was used. A similar methodology was used in the dissertation [14], where various types of HMMs were also used to diagnose and predict drill-bit conditions. This dissertation also makes use of support vector machines to classify sequential data. In [15] HSMMs are used to create a similar framework, using several HSMMs in sequence to diagnose and predict the state of degradation of hydraulic pumps.

Many studies done on predictive maintenance using ANNs emphasise the ability of these networks to accurately diagnose and predict future states. In [16], a simple neural network using signal values sampled at discrete times was used to estimate RUL, taking the age of the monitored equipment into consideration. In [17] and [18], neural networks were used to compute and continuously update distributions of RUL predictions based on vibration data from bearings. An attempt at a real-time implementation of condition-based monitoring of airport vehicles was done in [19], where several kinds of network architectures were implemented and compared. In [20] a special kind of recurrent neural networks, known as Elman-networks, were used to predict the behaviour of metal cutting tools.

Articles where methods that are very different from those above are also plentiful. The authors of [21] introduce a criticality (CRIT) index, which they use to determine the current status of a system and to estimate the remaining life of a system based on how the system is used. In [22], logistic regression is combined with an auto regressive moving average (ARMA) model to analyse the behaviour of elevator doors, both with and without access to historical data. An additional approach is given in [23], where a model of system reliability is used, together with collected condition data, to estimate the RUL of a steel mill.

In many of the articles exploring predictive maintenance cited above, very good predictive and diagnostic results are reported. However, these results are often obtained on rather small systems, and experiments are able to be executed under controlled conditions. Exceptions to this are for example [19] and [24], where case studies are carried out on airport vehicle doors and train bogies respectively. These studies highlight some of the problems which occur when moving from small and controlled systems to large, real world systems. These problems include lacking system data, system modelling issues, and environmental impacts. Finally, [25]

gives a good introduction to various areas where predictive maintenance have been used, as well as a short review of open research questions in the field.

1.4 Thesis Outline

The remainder of this thesis is organised as follows. Chapter 2 outlines the theoretical ideas behind the methods used in this thesis. This includes a brief introduction to subjects in data mining such as itemset mining and association rule learning, data classification using boosted classification trees and the C5.0 algorithm, and finally autoregressive models for time series data. In Chapter 3 an overview of the data and the data exploration that was performed is given. Due to the reliance on data quality and quantity in later stages of the thesis, the results from the data exploration are also presented. These results motivate the method choices, which are then described in Chapter 4. Chapter 5 presents the results obtained by the diagnostic and prognostic attempts that were made, followed by discussion around these in Chapter 6. Lastly, Chapter 7 summarises the thesis, and also includes some future outlooks and suggestions for further work based on the results found in this thesis.

2

Theoretical Concepts

As the present chapter title implies, this chapter serves to give an overview of some of the theoretical concepts that underlie the methods used in this project. Some central concepts in the fields association rule learning, classification trees and autoregressive modelling are presented in sections 2.1, 2.2 and 2.3 respectively. While not a thorough review, this introduction is intended to give an overview of the theory that is relevant for this thesis work.

2.1 Frequent Itemset Mining and Association Rule Learning

In the field of data mining one of the key objectives is to find interesting and significant patterns between variables or objects in a dataset. One type of such patterns, known as *associations*, are of the form IF A THEN B. In such relations, the presence of some object, or objects, A, also implies the presence of some other object B. To formalise this notion, given a dataset $\mathbf{D}$ an *item* and *transaction* are defined as follows.

Definition 2.1: An *item* I is a binary attribute indicating the presence of an object in $\mathbf{D}$. $\mathcal{I} = \{I_1, I_2, ..., I_N\}$ denotes the set of N unique items which are present in $\mathbf{D}$. A *transaction* T is a binary vector, where the k:th element $T_k = 1$ if a corresponding item I_k is included in the transaction. The set $\mathcal{T} = \{T_1, T_2, ..., T_M\}$ is the set of all M transactions present in $\mathbf{D}$.

The dataset $\mathbf{D}$ consists of a number of unique transactions, each containing a subset of items, and we say that a collection of items X satisfies a transaction T if $T_k = 1$ for all items I_k in X [26]. Given these definitions we can in turn define an *association rule*, according to definition 2.2.

Definition 2.2: An *association rule* is an association of the form $X \implies Y$, where X is a subset of items in $\mathbf{D}$ and Y is another subset of items which are not present in X.

Borrowing terms from logic, X is said to be the antecedent of the rule, and Y is said to be the consequent of the rule. The goal in association rule learning is to find significant rules given a dataset of transactions. To determine which rules are significant a number of additional measures are used. Given two subsets of items X, Y, a dataset of transactions $\mathcal{T}$ and an association rule $X \implies Y$, the support, confidence and lift of X and $X \implies Y$ are defined as in the following three definitions.

Definition 2.3: The *support*, $\text{Supp}(X)$, of the itemset X is the proportion of transactions T in $\mathcal{T}$ which contain X,

$$\text{Supp}(X) = \frac{|T \in \mathcal{T}; X \subseteq T|}{|\mathcal{T}|},$$

where $|\mathcal{T}|$ denotes the size of the transaction dataset, *i.e.* the number of binary vectors in $\mathcal{T}$.

Definition 2.4: The *confidence* c of the rule $X \implies Y$ is the fraction of times which the rule is found to be true, *i.e.* the number of times that a transaction that includes X also includes Y,

$$c(X \implies Y) = \frac{\text{Supp}(X \cup Y)}{\text{Supp}(X)}.$$

Definition 2.5: The *lift* is a measure of how the observed support of the rule $X \implies Y$ compares to that expected if X and Y had been independent of each other;

$$\text{lift}(X \implies Y) = \frac{\text{Supp}(X \cup Y)}{\text{Supp}(X) \times \text{Supp}(Y)}.$$

Based on these definitions, significant rules are those which have a high confidence, indicating that they often hold true. The itemsets associated with these rules should also have a support exceeding some chosen support threshold. Itemsets satisfying this minimum support constraint are said to be frequent itemsets. However, given that there are n items in a dataset, there are $2^n - 1$ possible combinations of items that are candidate itemsets, as illustrated in figure 2.1. Given this exponential increase in the number of possible itemsets as the number of items in $\mathcal{I}$ increases, it soon becomes infeasible to generate all combinations of items. Instead, one can use algorithms exploiting the fact that most rules generated in an exhaustive manner will describe redundant relations. By ignoring such redundant itemsets, frequently occurring itemsets in the dataset can be found efficiently. One algorithm for this purpose is the *Apriori*-algorithm, which is described below.

2.1.1 The Apriori Algorithm

As introduced in [27], the Apriori algorithm uses a breadth-first-search approach to efficiently identify frequent itemsets in a dataset, given threshold levels for the

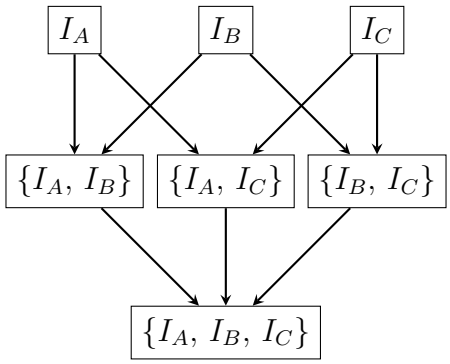


Figure 2.1: An illustration of the itemset tree that can be generated from an itemset of 3 items, $\mathcal{I} = \{I_A, I_B, I_C\}$. From this set there are $2^3 - 1 = 7$ possible itemsets which are part of the tree.

support and confidence levels of the resulting association rules. The key to the efficiency of the algorithm lies in its exploitation of the Apriori principle, which reduces the number of itemsets that needs to be generated. As presented in [28], the Apriori principle is given by the following proposition.

Proposition 2.1. *(The Apriori Principle)* If an itemset is frequent, then all of its subsets are also frequent. Likewise, if an itemset is infrequent, then all of its supersets are also infrequent.

For a proof of proposition 2.1, please refer to [29]. The Apriori algorithm utilises the implication of proposition 2.1 to prune the tree of item combinations whenever an itemset is found to be infrequent. It does this by iteratively creating itemsets of increasing size and then removing those found infrequent with regards to the minimum support threshold. This efficiently finds all significant itemsets in the dataset. In practice, a maximum itemset size can be specified to generate only itemsets below some maximum size, further limiting the number of itemsets evaluated. A pseudo-code outline of the Apriori algorithm, as implemented by [30], is given in algorithm 1.

Although efficient in reducing the search space of candidate itemsets, the Apriori algorithm does suffer from the fact that several passes need to be made over the data, one for each level of itemset size. For a large dataset, counting the frequencies of each itemset can also take a considerable amount of time. There exist algorithms based on Apriori which bypass these limitations. For example, the Eclat-algorithm uses depth-first-search and equivalence classes, and the FP-Growth algorithm uses a recursive divide and conquer methodology, These algorithms will not be covered in this thesis, and the reader is referred to [31, 32, 33] instead.

Algorithm 1: The Apriori Algorithm

Data: $\mathcal{T}$, a dataset of transactions t; s_{min}, minimum support; l_{max}, maximum itemset size.
Result: A list L of all frequent itemsets contained in $\mathcal{T}$, subject to the minimum support constraint s_{min} and size constraint l_{max}.

```
L_1 = All frequent itemsets of length 1;
while L_k ! = ∅ and k ≤ l_max do
    C_{k+1} = Candidate itemsets of length k + 1;
    foreach transaction t in T ∩ C_{k+1} do
        Increment the corresponding elements in C_{k+1};
    end
    L_{k+1} = Itemsets in C_{k+1} with support ≥ s_min;
    k + +;
end
return All frequent itemsets L_i, i = 1, 2, ...l_max
```

2.1.2 Association Rule Generation

Once the frequent itemsets in a transaction database have been found, the association rules relating these to each other need to be determined. To do this, one uses the support and confidence measure of the frequent itemsets found. Significant rules are identified as those which exceed a minimum confidence threshold c_{min}. A single pass over the entire set of frequent itemsets would find all rules fulfilling this constraint. However, one can use a method analogous to the pruning done when finding frequent itemsets in order to reduce the amount of rules considered. While pruning of infrequent itemsets is based on the fact that supersets of an infrequent itemset are also infrequent, the confidence measure of an association rule is not monotonous in this way. In general, this means that a the confidence $c(\{I_A, I_B, I_C\} \implies I_D)$ of a rule can be both higher or lower than $c(\{I_A, I_B, \} \implies I_D)$. However, as shown in [29], for rules generated from the same dataset it holds that

$$c(\{I_A, I_B, I_C\} \implies I_D) \geq c(\{I_A, I_B\} \implies \{I_C, I_D\}) \geq c(I_A \implies \{I_B, I_C, I_D\}). \tag{2.1}$$

Rooting the tree of possible association rules by the rule with the maximum number of items on the left hand side of the implication thus has a practical result. When doing this, all consequents of an association rule not satisfying the minimum confidence constraint can be pruned directly. This significantly reduces the number of rules needed to be investigated, as is illustrated in figure 2.2. The tree of itemsets can be traversed, storing high-confidence transitions along the way. By removing all branches starting in a node with confidence below the threshold c_{min}, all significant rules at this confidence level will be found[34]. Introducing pruning of low confidence branches, along with other optimisations of the algorithm implementation which are discussed in [27], can give speedups of orders of magnitude in finding significant association rules in the data. This enables much larger datasets to be efficiently processed.

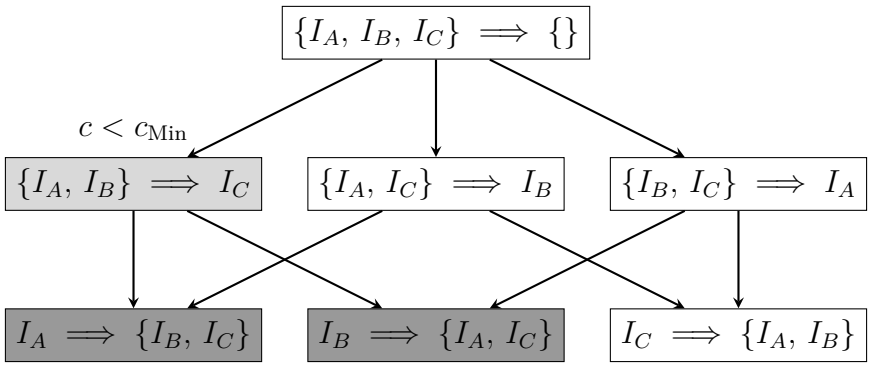


Figure 2.2: An illustration of the concept of association rule generation from an itemset of three items I_A, I_B, I_C. The shaded rule $\{I_A, I_B\} \implies I_C$ does not satisfy the confidence constraint, and hence it can be pruned from the tree, along with all its subrules.

2.2 Classification and Regression Tree Learning

Classification and regression trees, or CART, is a group of machine learning models with the common attribute that they are based on classifying data through sophisticated series of *if-else* statements. The basic structure of a CART model is that of a binary decision tree, which is anchored in a root node. Successive nodes are then added to the tree by performing binary splits based on the values of different input variables. This process results in a number of *branches* in the model. Given a target variable, a CART model attempts to either classify it or predict its value based on the other input variables. This is done by conditionally moving the input data along the branches of the model until a terminal node, holding the predicted value, is reached.

Consider a set of categorical training data with p input variables $(x_1, x_2, \ldots, x_p)$ and n observations, where each observation i belongs to one of k classes. The process of fitting a classification tree model consists of finding which of the input variables $(x_1, x_2, \ldots, x_p)$ to use for dividing the data into subsets, and also the threshold values where the splits should take place. The end goal is to maximise the homogenity, also called purity, of each class in the terminal nodes of the tree, as presented in [35]. In a perfect fit each terminal node of the classification tree would contain only observations from one of the k classes.

The number of splits made before a terminal node is reached is known as the depth of the tree. By increasing the depth of a decision tree, more precise classification criteria are defined. At the same time, however, increasing the depth leads to an increased risk of overfitting the model to the training data. Perfect purity could be obtained trivially by fitting a deep tree which simply places each observation in its own terminal node. However, a model fitted in such a way would be hopelessly dependent on the training data used in the fit, performing poorly when presented to previously unseen observations.

In order to reduce the risk of overfitting a classification tree model, various methods can be employed to increase prediction accuracy on new data. This process is known as pruning a tree, and works by either limiting the depth of the tree, or by removing branches from the model with little impact on predictive performance [36]. The former approach involves using a limit on model depth, preventing any more complex model from ever being created. In the latter approach the model is allowed to grow to maximum depth, after which branches are removed based on an error rate estimation criterion. An excellent review of the exact workings on these types of pruning can be found in [36].

2.2.1 C5.0 Decision Tree Generation

A popular algorithm for generating classification trees is the *C5.0* algorithm, as introduced in [37], originally developed by Ross Quinlan. When generating a model, C5.0 uses an information gain criterion to select the variable which will split samples into maximum purity subsets. The information gain is based on the concept of entropy from information theory. Given a set S of N observations, each belonging to one of c classes with probability p_i, $i = 1, 2, \ldots, c$. The entropy, or information content, of the class distribution is given by

$$H(S) = -\sum_{i=0}^{c} p_i \log_2 p_i. \tag{2.2}$$

When the base two logarithm is used in equation (2.2), the entropy H is measured in terms of *bits*. As shown in [38], entropy will be at its maximum ($H = log_2(c)$) in cases where each class occurs with equal probability, *i.e.* $p_1 = p_2 = \cdots = p_c = 1/c$. On the contrary, the entropy minimal and equal to zero when one class occurs with certainty.

Consider now the situation where the set S has been split into k subsets S_i, $i = 1, 2, \ldots, k$ by some criterion. The information content in the data before the split, H_{Before}, is given by the entropy of the class distribution as described by equation (2.2). The information content in the collection of subsets after the split is in turn given by

$$H_{\text{After}} = \sum_{i=1}^{k} \frac{n_i}{N} H\left(S_i\right), \tag{2.3}$$

where n_i is the number of observations placed in subset S_i after the split. The *information gain* (IG) from this split of the data is the difference in information content before and after the split;

$$\text{IG}_{\text{Split}} = H_{\text{Before}} - H_{\text{After}}. \tag{2.4}$$

A split resulting in a high information gain implies that either one or both of subsets obtained after the split contain a larger proportion of classes from one class than in the original data partition [39]. Thus, by seeking to maximise the information gain at each data division a splitting criterion is obtained.

One drawback of using the information gain criterion of equation (2.4) for partitioning a set of observations is that the criterion is biased towards splitting on variables taking many different values, as stated by [39]. To circumvent this bias the C5.0 algorithm does not use the information gain of equation (2.4) directly. Instead it uses a normalised information gain measure known as the *gain ratio*. Taking the information content after a split of the data into subsets to be

$$H_{\text{Split}} = -\sum_{i=1}^{k} \frac{n_i}{N} \log_2 \frac{n_i}{N}, \tag{2.5}$$

the gain ratio is then defined in [40] as

$$\text{GR}_{\text{Split}} = \frac{\text{IG}_{\text{Split}}}{H_{\text{Split}}}. \tag{2.6}$$

In addition to just maximising the gain ratio criterion of equation (2.6), the C5.0 algorithm further constrains which splits to make by the restriction that the information gain must be larger than the average gain over all splits considered, as described in [39].

When building a classification tree the C5.0 initially builds a full depth tree using the information gain criterion to decide whether to split the samples at each node or to leave the node as a terminal node [37]. Once the full, deep tree model has been fit it is pruned by either removing branches of the tree or by replacing an earlier node with a later branch. The decision of whether to prune a branch of the model is based on an estimation of the prediction error on new data both with and without the pruning. In [35], a confidence bound is used to determine if the pruning of a branch will significantly improve model accuracy. The branch is pruned from the model if this is the case.

2.2.2 Boosted Classification Trees

When training a single classifier using some training algorithm, such as the C5.0 algorithm described in section 2.2.1, it is often difficult to properly capture the behaviour of the underlying variable distributions when these follow a complex distribution. In such cases, using a large and complex classifier model can give good performance on training data, at the cost of overfitting. On the other hand, a simple model with good generalisation will fail to capture the behaviour in the data presented to the model. Either approach has shortcomings in terms of model accuracy and performance. An alternative approach to model fitting in this situation is the usage of so called *ensemble learning*. As presented in [41], the principle behind ensemble learning is to generate a number of simple classifiers, rather than a single complex one. Each classifier is then used to classify the data individually, after which the resulting classifications are combined to form a final prediction.

Several classes of ensemble learning are possible. In this thesis the focus is on so called *boosting*, which is a method implemented by, amongst others, the C5.0 algorithm. Due to the complexity of the theory behind boosting only a brief overview

focusing on the implementation of boosting in C5.0 will be given here. For a thorough review of boosting as a whole, refer to [42], which gives a comprehensive overview of the field.

In C5.0, boosting is implemented using a method similar to the *AdaBoost*, or *Adaptive Boosting* algorithm, which is presented in [42, 43]. This algorithm works by iteratively generating simple classifiers, each with a classification ability only marginally better than random guessing. After fitting an initial classifier to the data, successive classifiers are fitted by sampling the training data in such a way that they are exposed to more data which the previous models have failed to classify. In this manner, each iteration of classifier learns to classify data on which the previous iterations have failed. This ensures both that the newer classifiers are trained on different data than previous ones, and also that more attention is placed on data which is difficult to classify. Once a predetermined number of boosting iterations have been performed, or when boosting no longer improves model performance, the trained simple classifiers are combined to form the final predictive model. When presented to a new sample, each simple classifier computes a confidence value for each class. The class confidence is then averaged over all classifiers, and the predicted class is the one with the highest confidence value, as described in [35].

While boosting often is beneficial for a predictive model, there are a few caveats when using it for model fitting. Firstly, if the simple classifiers are not trained on sufficiently different samples, the boosted model will not perform any better than a single classifier [41]. Secondly, although each individual classifier fitted during boosting is simple, the fact that many classifiers are trained tends to increase the total computational power required to train the classifier. Performing excessive iterations of boosting may also lead to overfitting, much like the unboosted model. This phenomenon is further discussed in [42]. Finally, using a boosted model to train a classifier on noisy data can be very inefficient, and may sometimes lead to a decrease in performance compared to an unboosted model on the same data, as argued in [43].

2.3 Autoregressive Time Series Modelling

Data generated by a system is commonly received in the form of a *time series*. Given such a series of data, it is of interest in many areas to analyse the statistical properties of the system. However, many statistical methods are based on the assumption of observations in the data set being statistically independent of each other. The consecutive sampling of time series data, however, means that successive observations often are highly correlated. This invalidates the underlying assumptions of many conventional statistical methods, as stated in [44]. This in turn creates a need for models which can handle correlated data. Models for this purpose fall into the field of time series analysis. Such analyses include using historical data to forecast future values in the time series, or modelling the data to determine underlying trends in observations.

One way of quantifying the dependence between successive observations in a time series is to look at the *autocovariance* of the data. The autocovariance gives a measure of the covariance of the data with itself, considering observations occurring at different times. It is defined through definition 2.6 below.

Definition 2.6: The *autocovariance* between data sampled at times t_1 and t_2 of a time series is given by

$$r(t_1, t_2) = \mathrm{Cov}(y_{t_1},\, y_{t_2}) = \mathrm{E}\left[(y_{t_1} - \mu_{t_1})(y_{t_2} - \mu_{t_2})\right] = \mathrm{E}\left[y_{t_1} y_{t_2}\right] - \mu_{t_1}\mu_{t_2},$$

where $\mathrm{E}[\cdot]$ is the expected value operator and μ_{t_1}, μ_{t_2} are the expected values $\mu_{t_1} = \mathrm{E}[y_{t_1}]$ and $\mu_{t_2} = \mathrm{E}[y_{t_2}]$ at times t_1 and t_2 respectively.

From definition 2.6 it follows that $r(t,\, t) = \mathrm{E}\left[(y_t - \mu_t)^2\right] = \mathrm{Var}(y_t)$. Moreover, for uncorrelated data, such as white noise, it holds that $r\left(y_{t_1}, y_{t_2}\right) = 0 \,\forall\, t_1 \neq t_2$.

When discussing time series data, one should distinguish between data generated by a *stationary* process and data generated by a *non-stationary* process. Stationary processes can be further divided into *strict sense* stationary and *wide sense* stationary processes.

Definition 2.7: A *strict sense stationary* process of a random variable $X_t = (x(t_1),\, x(t_2), \cdots, x(t_N))$ has a joint distribution function $F_X(X_t)$ which is invariant over any arbitrary timeshift τ, *i.e.* $F_X(X_t) = F_X(X_{t+\tau}) \,\forall\, t, \tau$.

For a process that is wide sense stationary, time invariant properties are required only for the mean and variance of the process. More precisely, wide sense stationarity of a random process X_t is determined through the following definition, found in [45].

Definition 2.8: A *wide sense stationary* stochastic process $X_t = \{x_t; t \in \mathbb{N}\}$ fulfils that: 1) the variance of the process is finite, $\mathrm{E}\left[X_t^2\right] < \infty$ and constant; 2) the mean of the process is constant over time, $\mu_t = \mu_{t+\tau} = \mu$ and 3) the covariance between two values in the process depend only on the time τ between them. This last condition implies that $\mathrm{E}\left[x_{t_1} x_{t_2}\right] = \mathrm{E}\left[x_{t_1+\tau} x_{t_2+\tau}\right]$.

In general, wide sense stationarity is not a sufficient condition for strict sense stationarity. However, for some common processes, such as Gaussian processes, the resulting distributions are uniquely determined by their mean and covariance. For such processes wide sense stationarity does indeed imply strictly stationary [44]. For the remainder of this thesis, stationarity of a process will refer to wide sense stationarity, unless otherwise stated.

From definition 2.8 it follows that if a time series is stationary the autocovariance of two observations separated by a timelag k, $r(t,\, t-k)$ is independent of the initial time t. Moreover, this definition also means that the expected value is constant for a stationary process, so that $\mu_t = \mu_s = \mu$. These two properties imply that the

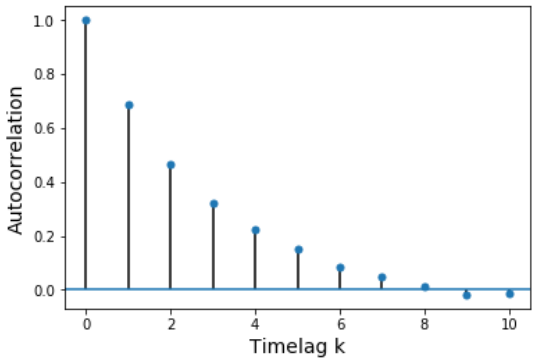
autocovariance can be redefined as a function dependent on the timelag k as

$$r_k = \mathrm{Cov}\left(y_t, y_{t-k}\right) = \mathrm{E}\left[y_t y_{t-k}\right] - \mu^2. \tag{2.7}$$

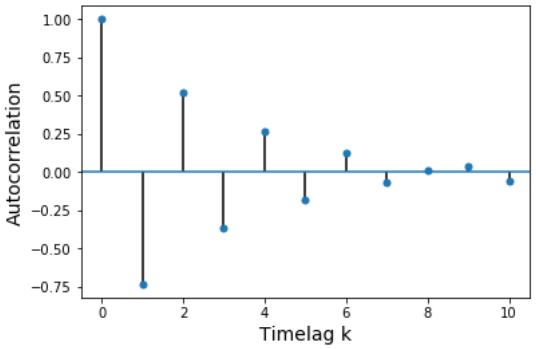
The autocovariance function of a time series has some useful properties. Firstly, $r_0 = \mathrm{Var}(y_t) \geq 0$; with equality only for constant, noise free data. Secondly, $|r_k| < r_0$; *i.e.* higher order covariances can never be larger than the variance in data. Finally, the autocovariance is symmetric, so that $r_{-k} = r_k$ [44]. This last property has the practical result that either positive or negative timelags can be considered for analysis. Related to the autocovariance is the *autocorrelation* $\rho(k)$ of a stationary process, given by

$$\rho_k = \mathrm{Corr}(y_t,\, k) = \frac{\mathrm{Cov}(y_t,\, y_{t-k})}{\mathrm{Var}(y_t)} = \frac{r_k}{r_0}. \tag{2.8}$$

The autocorrelation is useful for evaluating the relative importance of lagged values due to the fact that $|\rho_k| \leq 1$, with $\rho_0 = 1$. Thus, small values of the autocorrelation function for certain timelags k implies little influence on current values. An illustration of this can be seen in figure 2.3. This figure shows the autocorrelation at different timelags k of two data series, generated with either a positive- (2.3(a)) or a negative correlation (2.3(b)) with the previous value in the data series. Note how the autocorrelation is non-zero even over long timelags, despite no explicit dependence across lags greater than one. However, the influence decays rapidly over time. Also note how the negative dependence in figure 2.3b induces changes in the sign of the autocorrelation between successive timelags.



(a) Autocorrelation for timelags up to $k = 10$ for an AR(1) process generated by $y_t = 0.7y_{t-1} + \varepsilon_t$.



(b) Autocorrelation for timelags up to $k = 10$ for an AR(1) generated by $y_t = -0.7y_{t-1} + \varepsilon_t$.

Figure 2.3: A comparison of estimated autocorrelation at increasing time lags for positive (a) and negative (b) coefficient in an AR(1) process. In each case the autocorrelation is estimated from a sample of 2000 points generated by the process, and in both cases the noise term is given by $\varepsilon_t \sim \mathrm{N}(0, \sigma^2)$, $\sigma^2 = 0.2$.

A specific type of models for stationary time series data are *autoregressive*, or AR-, models. These models seek to capture data behaviour where new values are linearly dependent on previous values. As such, an autoregressive model is essentially a regression of new values in the time series onto previous ones. The number of previous values included in the model is called the *order* of the model, and is denoted by p. Naturally, a higher degree p results in a more complex model, as older values are given influence in the model. Formally, an autoregressive model is defined as follows.

Definition 2.9: Consider a stationary time series of data $\{y_1, y_2, ..., y_T\}$ with samples at times $t = (1, 2, ..., T)$. An autoregressive process of order p, denoted by AR(p), is then defined through

$$y_t = \sum_{k=1}^{p} \phi_k y_{t-k} + \varepsilon_t,$$

where ϕ_k are the model parameters and ε_t is a random noise term, representing the irreducible error in the data. This random noise is assumed to be Gaussian white noise, $\varepsilon_t \sim \mathrm{N}(0, \sigma^2)$.

An autoregressive process can also be defined in terms of a *shift operator* B, as

$$By_t = y_{t-1}, B^2 y_t = y_{t-2}, \cdots, B^k y_t = y_{t-k}. \tag{2.9}$$

Using the shift operator an AR(p)-process is defined [44] through

$$\psi(B)y_t \equiv (1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p)y_t = \varepsilon_t. \tag{2.10}$$

The formulation in terms of the shift operator as in equation(2.10) gives a rather unintuitive interpretation of an AR process. By rearranging equation (2.10) as

$$y_t = \frac{1}{\psi(B)}\varepsilon_t = \psi^{-1}(B)\varepsilon_t \tag{2.11}$$

an autoregressive process can be seen as the result of a linear filter having the transfer function $\Psi(B) = \psi^{-1}(B)$ being applied to a signal ε_t of white noise, as described in [46]. This interpretation is helpful for understanding the notion of process stability, which is the topic of section 2.3.2.

2.3.1 Parameter Estimation Using the Normal Equations

Consider a stationary autoregressive process of order p, *i.e.* where

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ ... \ + \phi_p y_{t-p} + \varepsilon_t. \tag{2.12}$$

Assuming that the data y_t, $t = 1, 2, ..., T$ has been observed, the model coefficients ϕ_i, $i = 1, 2, ..., p$ are to be determined. One way of estimating these parameters is by using the *normal-*, or *Yule-Walker* equations. These equations are derived as

follows. Firstly, equation (2.12) is multiplied by y_{t-k} on both sides, resulting in the new equation

$$y_{t-k}y_t = \phi_1 y_{t-k}y_{t-1} + \phi_2 y_{t-k}y_{t-2} + \ ... \ + \phi_p y_{t-k}y_{t-p} + y_{t-k}\varepsilon_t. \tag{2.13}$$

Next, taking the expected value of each side of (2.13) results in

$$\begin{aligned} \mathrm{E}\left[y_{t-k}y_t\right] &= \mathrm{E}\left[\phi_1 y_{t-k}y_{t-1} + \phi_2 y_{t-k}y_{t-2} + \ ... \ + \phi_p y_{t-k}y_{t-p} + y_{t-k}\varepsilon_t\right] \\ \mathrm{E}\left[y_{t-k}y_t\right] &= \phi_1 \mathrm{E}[y_{t-k}y_{t-1}] + \phi_2 \mathrm{E}[y_{t-k}y_{t-2}] + \ ... \ + \phi_p \mathrm{E}[y_{t-k}y_{t-p}], \end{aligned} \tag{2.14}$$

where the second equality results from the assumption that the noise ε_t and y_{t-k} are uncorrelated, meaning that $\mathrm{E}[y_{t-k}\varepsilon_t] = 0$. Observing that $\mathrm{E}\left[y_{t-k}y_t\right]$ is the covariance between y_t and y_{t-k}, equation (2.14) can be rewritten as

$$r_k = \phi_1 r_{k-1} + \phi_2 r_{k-2} + \ \cdots \ + \phi_p r_{k-p}. \tag{2.15}$$

Note that equation (2.14) is similar to the original equation (2.12), but given in terms of the autocovariance r_{k-i} rather than the observed values y_{t-i}. Now, substituting $k = 1, 2, ..., p$ and utilising that $r_o = 1$ in (2.15) results [46] in the equation system

$$\begin{aligned} r_1 &= \quad \phi_1 + \quad \phi_2 r_1 + \cdots + \phi_p r_{p-1} \\ r_2 &= \phi_1 r_1 + \quad \phi_2 + \cdots + \phi_p r_{p-2} \\ &\vdots \qquad \vdots \qquad \vdots \quad \cdots \qquad \vdots \\ r_p &= \phi_1 r_{p-1} + \phi_2 r_{p-2} + \cdots + \quad \phi_p \end{aligned} \tag{2.16}$$

which constitutes the aforementioned *Yule-Walker* equations. These equations can in turn be rewritten in vector from as $\mathbf{r}_p = \mathbf{R}_p \phi$, where

$$\mathbf{r}_p = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_p \end{bmatrix}, \quad \mathbf{R}_p = \begin{bmatrix} 1 & r_1 & r_2 & \cdots & r_{p-1} \\ r_1 & 1 & r_1 & \cdots & r_{p-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{p-1} & r_{p-2} & r_{p-3} & \cdots & 1 \end{bmatrix}, \quad \phi = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_p \end{bmatrix}. \tag{2.17}$$

Using this vector notation the coefficients of an AR(p)-model can be obtained through

$$\boldsymbol{\phi} = [\phi_1, \phi_2, \ldots, \phi_p]^T = \mathbf{R}_p^{-1}\mathbf{r}_p. \tag{2.18}$$

The autocovariance matrix $\mathbf{R}_p$ of equation (2.17) is in the form of a symmetric *Toeplitz matrix*, where all diagonals take constant values and $\mathbf{R}_p = \mathbf{R}_p^T$. This is similar [47] with the covariance matrix of a general stochastic vector $\mathbf{x} = [x_0, x_1, \cdots, x_N]$, which is of the form

$$\mathrm{Cov}\left(\mathbf{x}\right) = \mathrm{E}\left[\mathbf{x}^T\mathbf{x}\right] = \mathrm{E}\begin{bmatrix} x_0x_0 & x_0x_1 & \cdots & x_0x_N \\ x_1x_0 & x_1x_1 & \cdots & x_1x_N \\ \vdots & \vdots & \ddots & \vdots \\ x_Nx_0 & x_Nx_1 & \cdots & x_Nx_N \end{bmatrix} = \begin{bmatrix} r_0 & r_1 & \cdots & r_N \\ r_1 & r_0 & \cdots & r_{N-1} \\ \vdots & \vdots & \ddots & \vdots \\ r_N & r_{N-1} & \cdots & r_0 \end{bmatrix}.$$

In general, only a finite series of samples from a stochastic process are available for analysis. As such, the exact value of the covariances in equations (2.17) and (2.18)

are unavailable and have to be estimated from data. According to [48], for a series of data with N samples, $\{y_1, y_2, \cdots, y_N\}$, estimates of the mean and autocovariance of the data are given

$$\begin{aligned} \hat{\mu} &= \frac{1}{N}\sum_{t=1}^{N} y_t \\ \hat{r}_k &= \frac{1}{N}\sum_{t=k}^{N} (y_t - \hat{\mu})\,(y_{t-k} - \hat{\mu})\,. \end{aligned} \tag{2.19}$$

Typically, as stated in [46], given a sample size N it is reasonable to estimate the covariance for timelags up to $k \approx N/4$.

2.3.2 Stability of Autoregressive Models

At the start of section 2.3 the notion of a stationary stochastic process was introduced. When fitting an AR(p) model to data, many useful properties arise if the values generated by the process are stationary for all times. Considering that an AR model can be seen as a linear system, as in equation (2.11), such stationarity can be expressed as a bounded input, bounded output stability criterion, as shown in [49]. If such a condition is fulfilled, a finite input to the system results in a corresponding finite output. For an AR(p) process the current value is recursively dependent on previous values. Taking an AR(1) process as an example, the output y_t for arbitrarily long timelags k is given by

$$y_t = \sum_{k=0}^{\infty} \phi_1^k y_{t-k}, \tag{2.20}$$

implying that $|\phi_1| < 1$ if y_t is to be bounded [48]. However, for a general AR(p) processes of arbitrary order p, this recursion quickly becomes very complex. Stability of such a process can instead be determined by looking at it through the linear system form of equation (2.11). Such a system has an all pole system function $H(z)$ on the form

$$H(z) = \frac{1}{1 + \sum_{i=0}^{p} \phi_i z^{-i}}, \tag{2.21}$$

where z is a complex valued variable. Assuming for simplicity that the corresponding poles of (2.21) are distinct, this function can in turn be factored as

$$H(z) = \frac{z^M}{(z - z_1)(z - z_2)\cdots(z - z_p)}, \tag{2.22}$$

where z_i is the i:th pole of the system. Equation (2.22) can in turn be expressed in terms of a partial fraction decomposition,

$$H(z) = \sum_{i=1}^{p} a_i \frac{1}{z - z_i}, \tag{2.23}$$

where a_i are constant coefficients. If the system is to be stable, it is clearly required that all (complex) poles z_i of (2.23) have a magnitude less than one, *i.e.* $|z_i| < 1$ [46]. In addition to just determining the stability of the model, the placement of the poles in the complex plane also gives an insight into the behaviour of the system, as described in [50].

3

Data Exploration

This chapter presents the initial data exploration of the provided data. This process was performed to familiarise with the data, and to determine which methods would be good choices for handling the data during later parts of the project.

3.1 Data Collection

All the data used in this thesis was supplied by AB Volvo. It consisted of three separate datasets, containing data collected from trucks operating in Sweden during 2016. Most of the provided data was contained in a file consisting of logs of status data uploaded from trucks. Data entries were uploaded either via a telematic uplink or through Volvo's TechTool2 system. Data uploaded via TechTool2 is collected when trucks are taken in for service, while telematics data can be uploaded remotely while a truck is in service. Each entry in this data consisted of a dated reading of some parameter in a truck. There is a large variety of parameters being logged for different purposes. Some of these are used by Volvo for monitoring the condition of their trucks, whereas others are used for performance tracking, or are legally regulated by ISO standards to be logged. With each upload from a truck, a bunch of entries of different variables are made in the database. These uploads forms a time series of data entries from each truck, which shows how parameter values change, allowing for truck condition can be monitored over time.

The two remaining datasets contained details of service and replacements carried out on all trucks, and specifications of all trucks included in the data. Entries in this data consisted of different types of service and repairs, along with information related to the service. Each type of repair made on a truck was given a separate entry, resulting in groups of data entries from the same service occasion. Each entry was also dated and had several identifiers, allowing separate service visits of a specific truck to be identified. Since this service data contained information on repairs and replacements made on trucks, it was intended to be used for relating truck condition, as monitored by the parameter logs in the log data, with repairs made to trucks.

For the remainder of this thesis these three datasets will be referred to as the "log data", "service data" and the "truck data" respectively. The truck data data was used only as a look-up table to identify the different trucks and not used for any direct analysis, while the log and service files were used as data sources for analysis. Therefore only the latter two datasets will be further presented in this chapter.

3.2 Data Exploration

Before any closer analysis of the data could be made it was necessary to familiarise with it in order to determine its structure and contents. Due to the data dependency of this thesis project, a good understanding of the available data was required in order to know which methods would be best suited for the end purpose of predictive analysis. Owing to the vastly different size and contents of the service- and log data these were explored separately.

3.2.1 Truck Service Data

Due to the relatively small filesize, only 200MB, of the service data no special structure for data storage was required and the data was handled directly in memory. The raw data consisted of approximately 1.3 million observations in 20 variables. A complete list of the variables present in the raw data can be found in table 3.1.

Variable selection was carried out by removing data columns associated with variables with little relevance for the purpose of predictive maintenance. Variables deemed to be irrelevant were those not directly related to the condition of the vehicles, or those which carried little information relating to what service had been carried out. Based on these conditions a total of 6 out of the original 20 variables were kept. Retained variables were the repair dates, to predict when repairs were carried out, and the part names to know which parts were repaired. Vehicle age and mileage were kept to have a measure of how much each vehicle was used. Both vehicle- and repair identification numbers were kept for organising the data. Common amongst the removed variables were that they either contained mostly missing or otherwise nonsense values (*e.g.* "serial number"), that they had no clear usage (*e.g.* "payment code"), that they contained duplicated information (*e.g.* "techla"), or some combination of these (*e.g.* "description").

Despite removing variable columns, a problem remained that many of the retained variables contained missing values, especially the "part name" field. Since this field was of high interest in the forthcoming analysis it was deemed best to remove entries containing missing values. However, doing so removed around 700,000 observations from the data, leaving just over 600,000 observations in the data.

Table 3.1: List of variables present in the provided service data.

Variable name	Data type	Notes
Vehicle ID	Integer	Vehicle identifier
Repair ID	Double	Identifier for repair occasion
Repair date	Timestamp	Time and date of repair
Part name	String	Name of part subject to repair
Part fgrp code	Integer	Related to part name
Techla	String	Related to part name
Quantity parts	Double	Quantity of parts used
Language code	String	Many values are missing or are unknown
Description	String	Swedish translation of variable 'Part name'
Mileage	Integer	Mileage at time of service, unknown unit
Vehicle age	Integer	Age at time of service, unknown unit
Repair order number	Integer	Unknown usage
Inv credit flag	Integer	Unknown usage
Visit reason code	String	Unusable due to missing values
Payment code	Integer	Unknown usage
Sequence number	Integer	Unknown usage
Serial number	Integer	Unknown usage, mostly *NULL* or *NA*
Job number	Integer	Unknown usage
Part prefix	String	Short string, unknown usage

3.2.2 Truck Log Data

The raw truck log data was provided in the form of a .csv-file with a size of just under 28 GB. The data consisted of roughly 185 million log entries, with each entry containing 16 recorded variables. The structure of the raw log data was inspected, allowing the rough structure of the data to be determined. This also enabled some basic inference of what information each variable contained. A summary of all variables in the data can be found in table 3.2.

The size of the raw log data file being too large to keep in computer memory at once made working with data slow in its original .csv format. Therefore, a more efficient way of accessing the data was required before closer analysis of the data could be made. In addition to its size, the way in which the data was presented in the log data file created a need to selectively access the data. The values reported in each data entry depended not only on the parameter code, but also on the three index levels. Only some of the combinations of index levels for each parameter code actually represented a logged value, whereas other combinations were only reported as constant bits. Since the latter type of data were of no interest in the intended predictive analysis a way was needed to extract only the data entries containing relevant numerical values.

Table 3.2: List of variables in the truck log data.

Variable name	Data type	Notes
Vehicle ID	Integer	Vehicle identifier
ECU parameter ID	Integer	Parameter code ID number
Parameter Code	String	Identifier for logged parameters
Reading ID	Integer	Identifier for data reading
Readout datetime	Timestamp	Time and date of readout
Readout date	Timestamp	Date of readout
Index level 0	Integer	Index bit
Index level 1	Integer	Index bit
Index level 2	Integer	Index bit
Value	Double	Value of parameter, given indices
Value string	String	Mostly missing (*NA*) values
Data type	String	Type of data, either 'Scalar' or 'Bit'
Bucket ID	Integer	Unknown usage
Sender	String	Data source
Type of dump	String	Automatic or manual readout
Send datetime	Timestamp	Time and date of data transfer

The choice of framework for data handling fell on using an infrastructure based on the Elastic Stack[1] open source software package. This framework consists of three modules used to search and visualise the data; *Logstash* which enables continuous parsing of an input data source; *Elasticsearch* which is an engine for searching and analysing data; and *Kibana*, which is used for visualisation and analytics of data processed by Elasticsearch. Using this framework enabled the data to be parsed from the original .csv-file and indexed into a searchable database. A more detailed description of the entire data indexing process is given below.

1. The raw .csv-file was parsed using Logstash into a local database. This parsing enabled a selection of specific rows columns of the data to be excluded in the output database, based on the inspection in the previous step. In this way only nine variables were included in the final database. Referring to table 3.2, the vehicle and reading identification numbers, readout date, type of sender, parameter code, the three index variables and the reading value were kept in the database.

2. The new database was processed using Elasticsearch, creating an index which allowed database queries to be made to access selected parts of the data.

3. The now indexed data was imported either into Python for numerical analysis, or into Kibana for visualisation purposes. In the former case, using the elasticsearch library in Python made it possible to import selected parts of the data for various types of analysis.

[1] `https://www.elastic.co/`

With an indexed database in place selected contents of the log data could be readily extracted and analysed. At this point the intent was to identify which parameters that were potentially useful for predictive analysis. The usefulness of a parameter was based on whether it was a potential indicator of the condition of some subsystem on a truck, or that a parameter corresponded to an actual fault of some sort. To do this, each parameter code and index combination had to be manually checked against a list containing parameter descriptions provided by Volvo. A total of 83 different parameters were logged in the data, each containing a varying number of sublevels, indexed by the three index variables. During the parameter identification process it was discovered that a vast majority of the provided data, almost 180 million out of the 185 million observations, were logs of a number of parameters corresponding to measurements of a "ranking index". After consultation with Volvo regarding what kind of measure this was it was dismissed as a usable predictor for the purposes of predictive maintenance. This resulted in 180 million observations being excluded from future data analysis, leaving approximately 5.5 million observations in the data.

Having had to remove a large portion of the provided data, a number of the remaining parameters were still identified as potentially good indicators of faults. These parameters included those related to coolant temperature, engine overload and clutch wear. Other parameters were found to be candidates as truck condition monitors, including engine run time, oil pressure and oil degradation. However, it was discovered that almost all the parameters identified as fault indicators only had observations for one truck. Having only one effective observation of the parameters related to faults, predicting when any of these had been triggered was not feasible using the data at hand.

Due to the lack of data which could be attributed to faults in the trucks, further usage of the log data was reevaluated. Having no diagnostic parameters available in the data, the objective of the predictive analysis was changed to be an investigation whether parameters related to truck usage could be accurately modelled and forecasted. The intention with this kind of modelling was to see if abnormal behaviour in these parameters could be detected and predicted.

With the new modelling objective in mind, parameters related to measurements made over time were considered to be of interest. However, a problem of data continuity emerged, as is shown in figure 3.1. When the total number of readouts each week is shown over the year there are two distinct periods where there are no observations recorded in the data. Especially the second of these, spanning from late August to late November, leaves almost a quarter of the year completely void of any data points. Related to this issue was also the length of the series of data points for each truck. Even the parameters having the largest number of observations in the data only had a handful of observations per truck. This gives rise to issues with the estimation of data statistics, as very few data points are available for estimating statistics and model parameters.

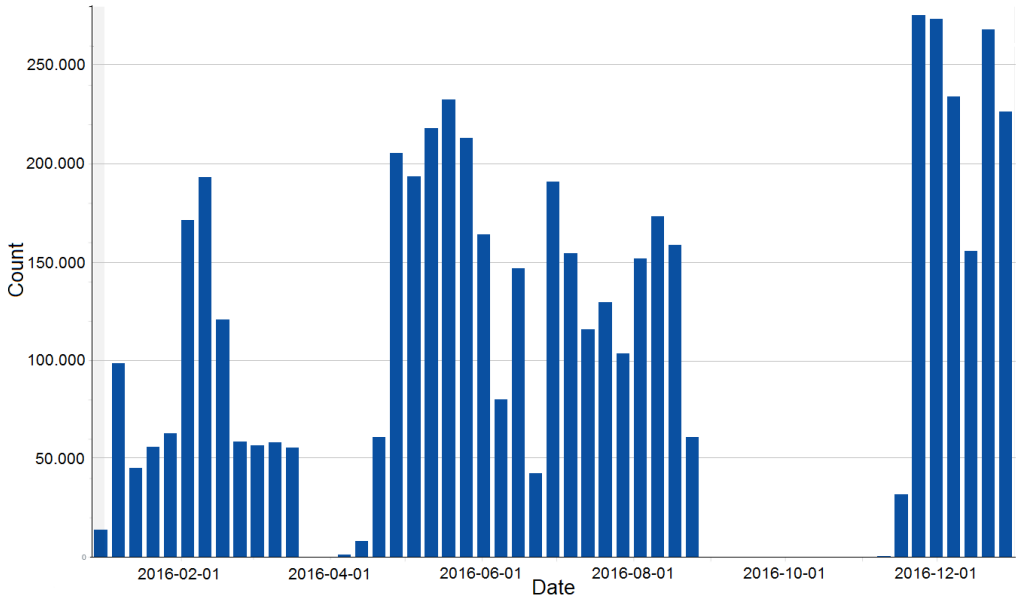


Figure 3.1: Histogram of weekly readouts of all parameters in the data after having removed observations which were logs of any of the "ranking index" parameters.

Another problem that was discovered was that of duplicate entries in the data, meaning that the number of unique observations was less than the total number of observations, and that not all parameters had been logged for every vehicle. The number of usable observations of each parameter was thus much less than in the raw data, and only a few of the parameters present in the data contained a reasonable number of observations per vehicle. Further discussion on this topic is given in section 6.1

4

Methods

Given the results of the data exploration performed in Chapter 3, a few different approaches were considered for analysing the data. As mentioned ealier, analysis was carried out separately on the service and the log data. The intention was to see if diagnostics of repairs and services could be made from the service data, and if prognostics of truck status could be made from the log data.

Diagnostics was carried out using two different approaches. First, data mining techniques were used to investigate whether there was any structure in the data which could potentially be exploited. This approach is described in section 4.1. Secondly, classification trees were implemented in an attempt to diagnose the presence of repairs based on other repairs carried out at the same time, and how this was done is presented in section 4.2. The approach taken when performing prognostics is introduced in section 4.3, which describes how autoregressive models were used to capture the behaviour of the time series log data from the trucks.

4.1 Mining the Service Data

The data mining techniques first used to analyse service data were frequent itemset mining and association rule learning. As described in sections 2.1 and 2.1.2 these are unsupervised methods, seeking to discover frequently occurring combinations of items in a database. From this, it can be inferred if a combination of some items can be associated with the presence of another item. There were two main goals when applying these methods to the data. The first was to see if there was any underlying structure in the service data which could be an indication of relations between different types of repairs. The second goal was to investigate if there were clusters of repairs which were typically carried out at the same time. The purpose of doing this was to see if there are tendencies in the data that different types of service either stand out from each other, or if they are somehow related. Finding such tendencies would be an indication that the structure of the data could perhaps be exploited for detecting the presence of different service types.

4.1.1 Frequent Itemset Mining

The service data was first analysed using techniques for association rule learning. The purpose was to look for patterns such as *if parts A and B have been repaired, then part C has likely been too.* This was done using tools available in the programming language R. The rule mining in itself was done using the R package `arules`[1][51], which contains an implementation of the Apriori algorithm originally made by Christian Borgelt [27]. However, the raw data was not suitable for directly applying the data mining algorithms, and had to be cleaned and prepared before further processing. This pre-processing was done using R and is outlined next.

As mentioned in section 3.2.1, one issue encountered in the service data was that there were a large number of missing entries. The raw data also contained several columns with, for this purpose, irrelevant information, as shown in table 3.1. Since the missing values occurred in the data column containing the service type, such entries were removed from the data. Likewise, all columns except for those containing the vehicle identification number, repair identification number and the service types were removed. This reduced the data from 20 variables to just three, making data handling significantly easier. The choice to keep these three variables were that the imminent association mining was intended to find associations between repairs carried out at the same time. Thus, it was considered unnecessary to retain variables not directly related to either the part replaced or the repair occasion. Next, observing that several replacement parts were present only a handful of time in the data, the data was filtered once more. This filtering was done by removing observations of service types occurring less than $n_{\text{Min}} = 10$ times in the entire dataset. This limit was arbitrarily chosen to remove extremely rare repairs, so that these would not be cluttering the analysis. This filtering also removed some data entries with corrupted data. At the same time, the data was also filtered for duplicated entries, which were taken to be entries where the same part was reported as replaced multiple times at the same service occasion. Finally, the data was aggregated into a number of generic categories in order to reduce the number of different parts, and to consolidate similar parts with only a few observations each. Groups of items considered "small parts", such as nuts, bolts, rivets etc. were then removed from the data, along with items considered "peripherals", such as decals, handsets, seats etc.

The data remaining after cleaning was prepared for itemset mining by grouping the data based on service occasion. This transformed the data into a transaction database, with each row being a transaction consisting of all repairs and replacements made at the same time. The change made to the data by this transformation can be seen in table 4.1. Note how the entries in the transaction list contains all parts sharing the same vehicle- and repair ID. A summary of the most commonly occurring types of service in the final data can be seen in figure 4.1. In this figure the 10 most common service types in terms of the fraction of transactions containing each item is shown. Once prepared, the Apriori algorithm was applied to the transaction list in order to mine for frequent itemsets and association rules. To prune

[1] `https://CRAN.R-project.org/package=arules`

Table 4.1: Repair data before (a) and after (b) having been transformed into a transaction database. Note that all ID entries are figurative, and the part names reflect part categories rather than specific parts. The TID in table (b) is the transaction ID, enumerating transactions.

(a) The three columns of the service data after cleaning.

Vehicle ID	Repair ID	Part name
1	391	Filters
1	391	Engine
1	391	Kit
2	57	Fenders
2	57	Doors

(b) Transaction database of truck repairs.

TID	Part names
1	Filters, Engine, Kit
2	Fenders, Doors, Covers
3	Filters, Oil system, Kit
4	Oil system, Pipes, Air system
5	Engine

the resulting rule list the minimum support was set to $\text{Supp}_{\text{Min}} = 10^{-4}$, and the minimum confidence was set to $c_{\text{Min}} = 0.7$. A number of different maximum itemset sizes involved in the association rules to be generated were tried, using sizes in the range 2-5. The upper limit on itemset size was set to prevent occasional transactions, consisting of many items, from having an disproportionately large impact on end results as the itemset size increased. This also helped to find more general rules rather than long, specific ones. The low confidence threshold was set due to the large number of items in the database. Due to this, even quite commonly occurring items were only being included in a small fraction of the total number of transactions.

The output from the Apriori program was mainly in the form of a R-object, containing the results of applying the Apriori algorithm to the data. This enabled the results to be accessed both as text output and as different plots of the generated rules. A good overview of visualisation methods for association rules can be found in [52, 53]. To provide an intuitive overview of the structure of the rules, a graph based visualisation approach was also used. This was done using the software *Gephi*, which allowed the generated association rules to be represented as a graph. In this graph associations were represented as nodes, with edges connecting parts included in each rule. Using force directed graph drawing algorithms [54] available in Gephi, such as Force Atlas 2 and OpenOrd [55], enabled the rule graph to be partitioned based on the number of connections between two rules. Thus, clusters were formed in groups of closely related repairs based on the number of mutual rules between them.

4.2 Classification and Prediction of Vehicle Service

One of the main purposes of predictive maintenance is to be able to predict service and repair requirements based on current system health conditions. For this purpose, it was investigated whether the data supplied for this project contained sufficient

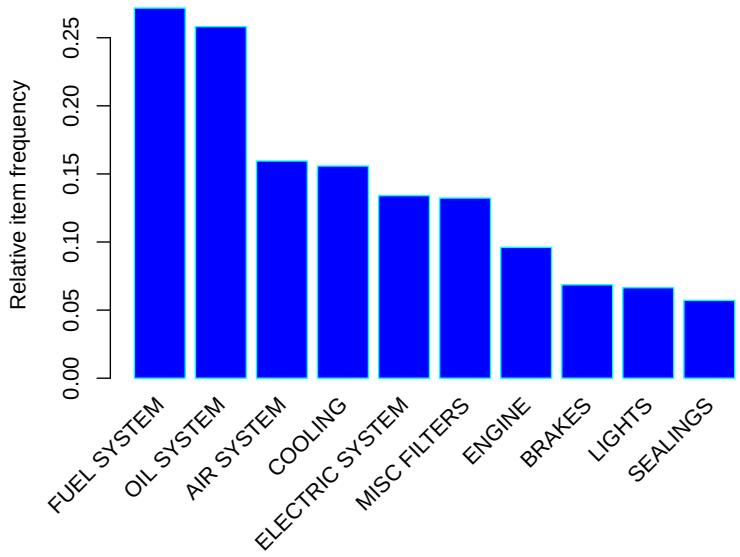


Figure 4.1: Frequency of the 10 most common items occurring in transactions in the data used for association rule mining.

information to be able to accurately predict when a certain type of service had been carried out on a truck. The intention was to see if a model for decision support could be made, which could serve as help to answer the question "If a truck is in condition A and we have repaired part B, is it a good idea to also check if part C is in need of service?"

The data used for this type of classification and prediction was also the service data. However, in order to be fit for the intended usage the raw service data was first subjected to significant preparation and augmentation before any further predictive analysis was performed. The following section will outline the pre-processing done on the data, after which the classification itself will be described.

4.2.1 Data Preparation

The service data was initially prepared much in the same way as when using it for frequent itemset mining as described in section 4.1.1. This meant that the raw data was parsed and filtered for corrupted entries and for repairs occurring less than $n_{\text{Min}} = 10$ times in the entire dataset, as well as removing duplicated or repeated entries. However, this time the mileage and vehicle age at the time of service were kept in the data, after having made a guess that these variables could potentially have a natural relation to the kind of service done to a truck. In addition, the data was augmented by also including the chassis type of each truck as a variable. This variable inclusion was also made based on a hypothesis that the type of chassis, indicating the type of usage of a truck, could influence the type and frequency of truck service.

Due to the large amount of service types present in the data, roughly 700 even after filtering away non frequent types, the raw data was also prepared by consolidating service types into a few, more general, groups. This process resulted in the 700 service types being reduced to only 43. The intention with this merging was to prevent problems with model selection and redundant variables during later classification. Another variable introduced to the data at this point was the time since the truck was last subjected to service. For those entries which constituted the first visit of the year for a truck this value was set to the average time between visits for all trucks.

With the raw data prepared, the next step of pre-processing regarded the structure of the data. This involved dummy coding the service events, based on the occasion at which service was carried out. This meant that the service types were introduced as dummy variables in the form of a co-occurrence matrix. This coding turned the dense 43 level variable of service types into 43 sparse, binary variables. The final dataframe resulting from the data preparation process consisted of roughly 78000 observations and a total of 47 variables. The variables retained, in addition to the 43 service indicator variables, were the age and mileage of each truck at the time of service, the truck chassis types and the time in days since the last service. A schematic view of this final data structure is seen in table 4.2.

Table 4.2: Structure of the data used for service prediction, consisting of a total of 47 variables of which 43 are the dummy coded service types.

Air system	Axles	$\cdots$	Windows	Mileage	Age	Chassis	Last service
1	0	$\cdots$	0	32859	230	FH	62
1	1	$\cdots$	0	49020	293	FH	61
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
0	0	$\cdots$	1	208	48	FM	117

4.2.2 Service Predictions

Without any obvious structure in the data to exploit for classifications the first step consisted of finding a suitable classification algorithm to use. The process of algorithm selection was performed as follows.

1. One type of service was selected as the target, *i.e.* the service which is to be classified as either present or not at a service visit. For the sake of algorithm selection the target was arbitrarily chosen as the first service variable, the trucks' air system. Also, only data entries where more than one part had been replaced at the same time were used, in an attempt to find primarily dependencies between service types.

2. The data was split into a training set, for fitting the predictive models, and a testing set for evaluating performance of the resulting model. 80% of the data

remaining after step 1 was used for training, leaving 20% for testing. The split into training and test data was done using the `caret` software package in R. This allowed for splitting the data into partitions based on class frequency of the target variable, where the class of an observation is either *service* or *no service*. Thus the proportion of service to non service observations in both the training and test data were the same as in the original data.

3. A selection of methods were used to fit a predictive model to the training data and to evaluate the predictive performance of the fitted models. Once again the `caret` package was used, allowing different classifiers to be trained using a similar model interface. The models tested were

   - Classification and regression trees (CART)
   - Random forest classification
   - Gradient boosted decision trees (GB Trees)
   - C5.0 decision trees
   - Support vector machine (SVM) using a radial basis function

   Each model was fit without any specific model tuning and using `caret's` default settings for each algorithm. Once fitted, the predictive performance of each model was evaluated on the test set. The only performance metrics considered at this point were the accuracy and Cohen's kappa statistic of each model, and is summarised in table 4.3. Although the accuracy is a quite crude performance metric, as described in appendix A, it was taken to be an indication of overall performance of each model. The kappa statistic was included to give a sense of how well the models classified each class, *i.e.* either service or no service.

Based on the performance of each model in step 3 it was decided to continue using C5.0 decision trees for further predictive analysis. This decision was based primarily on the pure predictive performance. The C5.0 model outperformed all other models in both metrics considered, as can be seen in table 4.3. Continuing to use the C5.0 algorithm to fit decision tree models, the next step was to test predictive performance for different types of service as targets. Additional tuning of the model was performed, to see if model performance could be further improved.

Table 4.3: Predictive accuracy and kappa statistic of the different models fitted to the prepared service data using "air system" as target variable.

Model type	Accuracy (%)	Kappa
CART	70.60	0.1654
Random forest	77.48	0.3902
GB Trees	73.03	0.266
C5.0	79.79	0.4608
SVM	72.63	0.1937

Due to time and resource constraints not all 43 types of service included in the data were used as targets. Instead, a few types were selected as being representative of "severe" and "standard" service procedures. Severe service procedures were those which were believed to be related to parts with critical functions or where a fault could lead to a loss of vehicle functionality. Standard service procedures were taken to be those which would either be expected to be routinely maintained, or where a failure would not cause any major malfunctions in a truck. Three service types were selected for each category; service related to the brakes, fuel system and the transmission were taken as severe faults, while service related to the electrics-, air- and oil systems were considered to be standard maintenance.

For the six selected fault types a baseline performance measure was established by fitting a classification tree using C5.0 with standard settings, without any boosting or misclassification penalisation. This process was intended to give an overview of how well the model performed over a range of target parameters. After this, one target was selected from each of the two fault categories and a new model was fine tuned to see if the predictive performance could be improved and, if so, to which degree improvements could be made. The fault types used for model optimisation were faults related to the oil system and faults related to the transmission of the trucks.

Optimisation of the classification model was carried out with regards to two model parameters. These parameters were the cost of misclassification and the number of boosting iterations used in the model. This was done using a tuning procedure, where a training grid of parameter values for evaluating model performance for different weight and boosting combinations was used. A total of 9 different values of false negative prediction cost were used, together with 13 different values for the number of boosting iterations. The values used can be seen in table 4.4.

Table 4.4: Parameter values used when evaluating the performance of the classification tree model fitted to the prepared service data. The leftmost values correspond to a baseline model with no misclassification cost or model boosting.

Boosting iterations	1	5	10	15	$\cdots$	55	60
False negative cost	1.0	1.25	1.5	1.75	$\cdots$	2.75	3.0

Given a target variable, in the form of one type of service, the data was split up into 10 folds. The data in each fold was sampled using stratified sampling to preserve the distribution of service/no service cases of the target variable in each fold. Cross validation was then performed for each combination of parameter settings in the training grid. This meant that a model was trained on nine of the data folds, using the current training grid parameters. Model performance was then evaluated on the last data fold, resulting in a number of model performance metrics; accuracy, kappa, sensitivity, specificity and balanced accuracy. An overview of these metrics can be found in appendix A. Model fitting was then repeated ten times, allowing each data fold to act as testing data. The training parameters were then updated along the training grid, repeating the cross validation of model performance for each combination of model parameters.

Once all points on the training grid had been tested, the model performance was averaged across all folds. This gave an estimation of overall model performance, based on the five performance metrics evaluated for each setting. For evaluating optimal performance, the model settings which maximised the kappa metric were used. The data was split a final time, fitting the best tuned model using 80% of the data. The last 20% of data was left for evaluating predictive performance. The entire process was repeated for the two target variables chosen, with performance then being compared between each of them.

4.3 Time Series Modelling of Truck Log Data

As presented in section 3.2.2, following the initial phase of data exploration the truck log data was analysed to see whether it would be possible to predict future trends in the different parameter logs, and to associate any such with different repairs being made to the vehicles. Due to the issues also outlined in 3.2.2 this analysis was severely limited in its scope. Given these conditions in data, the intent of the time series modelling of logged parameters were primarily to see whether a stable autoregressive model could be fitted to the data under these circumstances.

The autoregressive modelling was done in Python, using the indexed database created using Elasticsearch to selectively access the data. Due to the convoluted data structure of the log data, using parameter codes and index levels to specify what information was stored in each data entry, only a single parameter value at a time was modelled. Two parameter code/index level combinations, believed to be of interest for monitoring vehicle performance and condition, were selected for modelling. The ones tested are summarised in table 4.5. A sample display of some data series for each of the two parameters can also be seen in figure 4.2 and 4.3. The choice of parameters used in the autoregressive modelling was limited primarily by the number of observations available of each parameter. A thorough discussion on the limited modelling choices in the data is given in section 6.1.1. The air dryer predictive mileage parameter was chosen as it was not a cumulative variable and reflected the intention of predicting future condition of a truck. The total engine runtime parameter was selected due to that it gave an indication of truck usage and should thus reflect the rate of wear on many other components. Both these parameters also had a reasonable amount of observations. Given the parameter codes and the

Table 4.5: Description of parameters used in for autoregressive modelling of truck log data. Note that the parameter codes are anonymised.

Parameter	Description
Air dryer predictive mileage	Predicted mileage of truck air dryer, based on unknown measurements
Total engine runtime	Accumulated engine runtime since unknown starting time

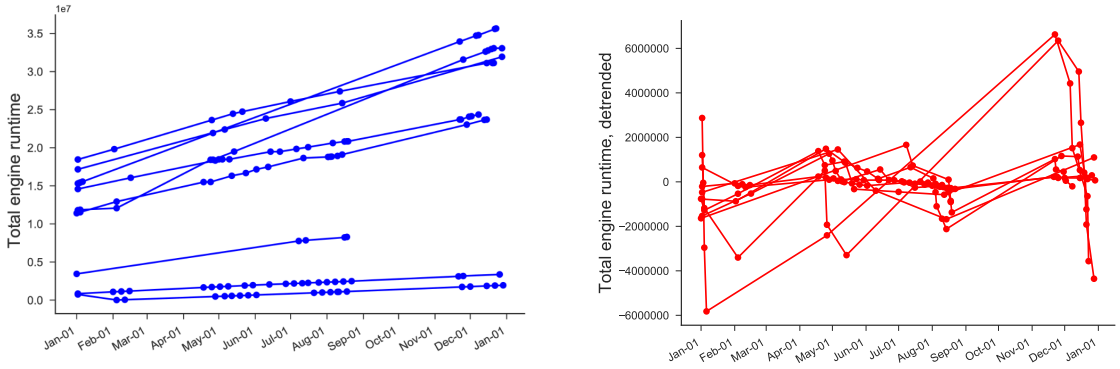


Figure 4.2: A sample of the first 10 data series of log values for the total engine runtime parameter. The left shows the original data series, while the right shows the same data after detrending using a linear regression fit to the data. Data points are marked with dots, and only data series with at least 4 datapoints are shown.

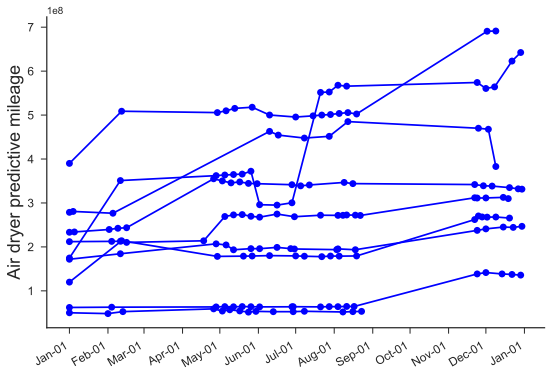


Figure 4.3: Sample plot of the first 10 data series of the air dryer predictive mileage parameter. As in figure 4.2, only data series with more than 4 entries are shown, with sample points marked as dots.

three index levels required to specify which parameter values that were sought, the autoregressive modelling was carried out as outlined in algorithm 2.

A closer description of the major steps in algorithm 2 is as follows. Lines 2-8 involve searching the database of truck logs for entries which match the specified parameter values. In this step the data handling framework based on Elasticsearch was used in order to efficiently search for the matching entries in the database. If there were any entries found matching the search query these were loaded into a dataframe in Python, in the process removing duplicated entries and sorting entries in order of vehicle identification number and readout date of the data entry. Since the time series data for each truck was to be modelled individually a list of all unique vehicles present in the data, as determined by the vehicle identification number, was also extracted from the search results.

Algorithm 2: Method for autoregressive modelling of truck log data

input: Database of truck log data
Parameter code of interest
Index levels I_0, I_1 and I_2
Autoregressive model order m

begin
  Search database for entries matching the specified parameter code and index levels
  if *Number of hits in database = 0* **then**
    Stop execution and report no entries in data
  else
    // There is data present, attempt modelling
    Load search results into Python
    Remove duplicate entries in search results
    V = List of all unique vehicles in data
    foreach *Vehicle v in V* **do**
      L_v = List of all data entries containing v
      $n_v = |L_v|$ // The number of entries in L_v
      if $n_v > 2p$ **then**
        // Only attempt to model time series with enough data
        (Detrend and scale L_v)
        $\mathbf{r}_p$ = Autocovariance(L_v, m)
        // Estimate model parameters using Yule Walker equations
        $\mathbf{R_p}$ = Autocovariance matrix // As per eq.(2.17)
        $\boldsymbol{\phi} = [\phi_1, \ldots \phi_p] = \mathbf{R}_p^{-1}\mathbf{r}_p$
        // Calculate system poles of equation (2.23)
        $\mathbf{z_0} = [z_{0,1}, z_{0,2}, ..., z_{0,p}] = Roots\,(H(z) = 0)$
      end
    end
  end
end

Lines 9-19 constitutes the main script loop for performing the autoregressive model fitting, looping over the list of unique vehicles in the data. For each vehicle in the list, all entries from this vehicle were isolated to form a time series of readings over the year. The number of entries, compared to the autoregressive model size p to be fitted, was then checked to see if modelling of the current data series was feasible. In order to have at least minimal support of the model fitted it was required that the underlying data series have at least $2p$ observations. The low margin in terms of the number of samples used for modelling meant that models were fitted without much support for the underlying statistics such as the covariance estimate. Referring to the recommendation of estimating no more than $N/4$ parameters using N samples in a series it can already here be flagged for potential modelling issues of data series with few observations.

If enough data was available in a series, *i.e.* the number of datapoints are at least twice the model size, two models were made. In the first model the data was used in its raw condition during the autoregressive modelling. For the second model the time series data was detrended, and normalised to unit variance. Detrending was done by fitting a linear regression model to the time series and then subtracting this fit from the values. This resulted in the detrended autoregressive modelling capturing expected deviations from a "normal" increase of parameter values over time, rather than the absolute values of the data. Detrending by taking delta values between datapoints was also considered, but due to the uneven sampling times this method was discarded.

An autoregressive model of order p was fitted to each data series by solving the Yule-Walker equations as described in section 2.3.1. In practice, due to the low number of datapoints in each vehicle time series, a model size of $p = 2$ was used. The poles to the linear system model description of (2.11) were also calculated to determine if the fitted model was stable or not. This process was then repeated for the remaining vehicles present in the data. The model parameters and characteristic roots were stored for each vehicle. These were then compared to see whether the fitted models were consistent between vehicles.

Evaluation of the autoregressive model fitting was carried out in two ways. The coefficient values of the fitted AR models were plotted as a histogram to give a view of the coefficient distribution. The characteristic root locations were also plotted as a heatmap in order to see if they were placed closely together. Both these evaluation methods were meant to see if the fitted model coefficients were consistent between trucks, and to investigate the model behaviour across the ensemble of trucks. In-sample predictions were also made to see how well the models captured the time series they were fitted from. This was done by taking the fitted model coefficients for each data series and then performing one step forward predictions, using the first data points in each data series to start the autoregressive recursion. The resulting data series were plotted together with the original data to see how well the original data and the predicted values followed each other.

5

Results

This chapter presents the results found during the different parts of this project. Section 5.1 regards the association rules discovered when mining the service data using the Apriori algorithm. The results are presented both as tables exemplifying some rules found, as well as graphs of the connections between service types and association rules. Section 5.2 then presents the results of predicting the occurrence of service types. Baseline results from fitting a classification model using several different target parameters are given, as well as the results when tuning the model to improve performance on two targets. Finally section 5.3 presents results obtained when modelling truck log data using an autoregressive model.

5.1 Itemset Mining in Service Data

Applying the Apriori program as described in section 4.1 to the service data, a large number of association rules were generated. Table 5.1 shows how many rules were generated for different maximum itemset sizes and support thresholds, given a minimum rule confidence of $c_{\text{Min}} = 0.7$.

Table 5.1: Table of the total number of rules generated using different maximum itemset sizes and support thresholds, with a constant confidence threshold of $c_{\text{Min}} = 0.7$.

Max itemset size	Support threshold	Number of rules
2	10^{-4}	7
2	10^{-5}	7
3	10^{-4}	508
3	10^{-5}	2495
4	10^{-4}	2880
4	10^{-5}	54319
5	10^{-4}	4894

To find the rules that stood out the most from what was expected, all rules were sorted by their lift value. The ten highest lift rules can be seen in table 5.2. Looking at the support of the discovered association rules, rather than the lift, the association rules the 10 highest support rules are recited in table 5.3.

Table 5.2: Top 10 association rules ordered by rule lift, using a maximum itemset size of 3, support threshold of 10^{-4} and minimum rule confidence 0.7.

Association rule			support	confidence	lift
{GRILLE,RADIATOR}	=>	{MEMBERS}	0.00012	0.75	1134.6
{LOCKS,MEMBERS}	=>	{GRILLE}	0.00012	0.86	400.3
{LOCKS,RADIATOR}	=>	{GRILLE}	0.00011	0.73	342.5
{ARM,GLASS}	=>	{FRAMES}	0.00012	0.75	236.8
{PLATES,TURBO}	=>	{SUPPORT}	0.00012	0.71	235.3
{ANCHORAGE,EXHAUST SYSTEM}	=>	{UNDERRUN GUARD}	0.00015	0.83	235.3
{MISC PIPES,TRANSMISSION}	=>	{HYDRAULICS}	0.00052	0.80	227.3
{HEAT SHIELD,WINDSHIELD}	=>	{UNDERRUN GUARD}	0.00011	0.79	221.8
{BUMPERS,SERVICE KIT}	=>	{UNDERRUN GUARD}	0.00013	0.76	215.9
{ANCHORAGE,BUMPERS}	=>	{UNDERRUN GUARD}	0.00031	0.74	208.4

Table 5.3: Top 10 rules ordered by support, obtained using a maximum itemset size of 3, support threshold of 10^{-4} and minimum rule confidence 0.7.

Association rule			support	confidence	lift
{OIL SYSTEM}	=>	{FUEL SYSTEM}	0.19974	0.8	2.8
{FUEL SYSTEM}	=>	{OIL SYSTEM}	0.19974	0.7	2.8
{ENGINE}	=>	{FUEL SYSTEM}	0.07077	0.7	2.7
{MISC FILTERS,OIL SYSTEM}	=>	{FUEL SYSTEM}	0.06568	0.9	3.3
{FUEL SYSTEM,MISC FILTERS}	=>	{OIL SYSTEM}	0.06568	0.8	3.2
{ENGINE,OIL SYSTEM}	=>	{FUEL SYSTEM}	0.05558	0.9	3.2
{ENGINE,FUEL SYSTEM}	=>	{OIL SYSTEM}	0.05558	0.8	3.0
{COOLING,OIL SYSTEM}	=>	{FUEL SYSTEM}	0.04908	0.8	3.1
{COOLING,FUEL SYSTEM}	=>	{OIL SYSTEM}	0.04908	0.8	2.9
{AIR SYSTEM,OIL SYSTEM}	=>	{FUEL SYSTEM}	0.04595	0.9	3.3

Due to the large amount of rules generated from the data it was not possible to obtain an overview of them by only looking at the text output from the Apriori program. As described in section 4.1.1 the generated rules were therefore also presented using a graph representation. This representation can be seen in figures 5.1-5.2. In these graphs the items and rules are represented as nodes An “item node” has an edge to a “rule node” if said item occurs on the left hand side of a rule. Likewise, a rule node has an edge to an item node if said item occurs on the left hand side of a rule. The nodes are placed in the graph based on the number of rules related to each item node. Rule nodes are clustered based on their connections with item nodes, so that mutual connections attract two nodes to each other while a lack of connection repels two nodes.

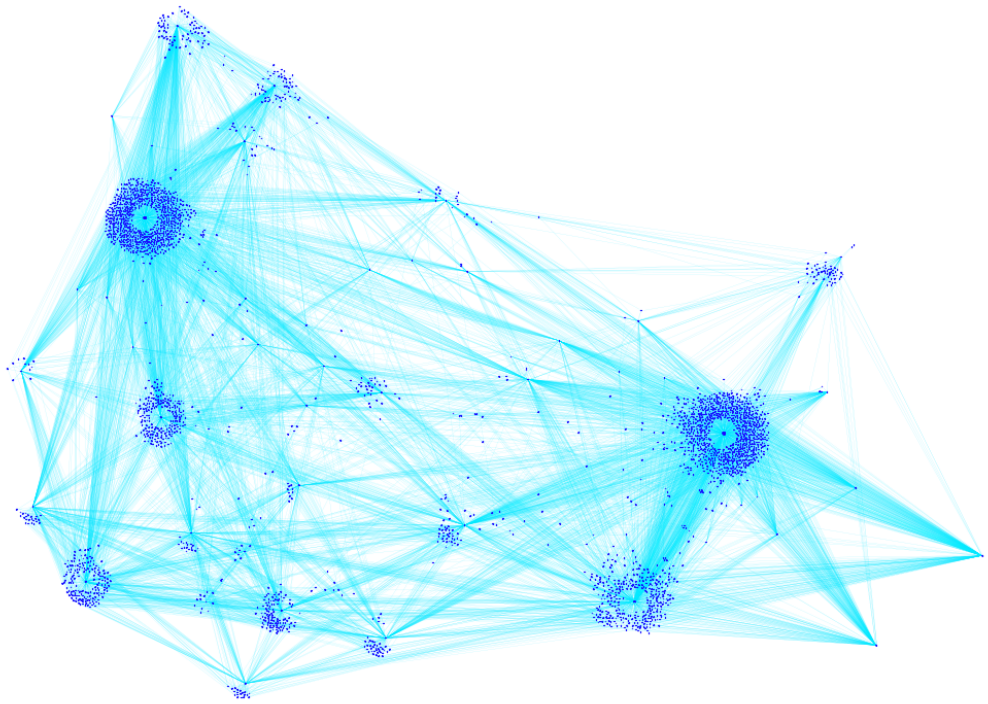


Figure 5.1: A graph representation, created using the ForceAtlas2 layout in Gephi, of the association rules found by using the Apriori algorithm. The maximum itemset size was set to 3, with support- and confidence thresholds of 10^{-4} and 0.7 respectively. In the graph the nodes represent service types and association rules, with two nodes being connected by an edge if they are part of the same rule.

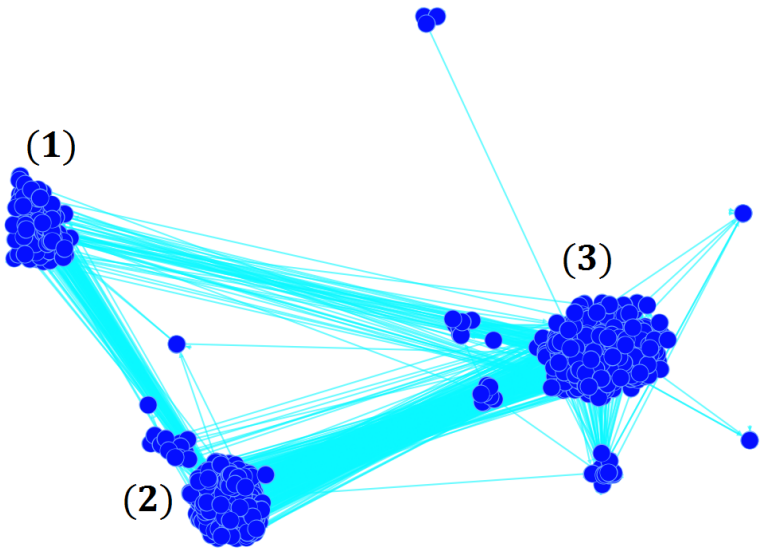


Figure 5.2: Graph of all generated association rules obtained using the OpenOrd force directed graph layout in Gephi, in order to emphasise clusters in the underlying ruleset. The maximum itemset size was set to 3, with support- and confidence thresholds set to 10^{-4} and 0.7 respectively. The numbers (1)-(3) indicate the three major clusters of nodes present in the graph, and are further discussed in 5.1.

5.2 Service Predictions

As described in section 4.2, predictive performance of the decision tree model fitted to the service data was evaluated across a grid of model parameters to find the best model fit. In 5.2.2 the metrics obtained over the training grid for the different target parameters used are reported.

5.2.1 Baseline Model Performance for Different Targets

To obtain a quantitative measurement of the performance difference that boosting and misclassification weights had on the predictive performance of the model, a baseline was made by fitting a model using neither boosting or weighting. The performance of this baseline model is summarised in tables 5.4 and 5.5. For a review of some of the metrics used in these tables, refer to appendix A.

Table 5.4: Performance of a baseline model, fitted with the C5.0 algorithm, using no boosting or misclassification weighting. A selection of different target variables were used, and for each one the model was allowed to train on 90% of the data, whereafter performance was evaluated on the remaining 10%. These splitting proportion of data was used to mimic the subsequent cross-validation splits.

Target	Accuracy	Kappa	Sensitivity	Specificity
Air system	0.7568	0.3264	0.2994	0.9716
Brakes	0.9418	0.2640	0.2003	0.9875
Electrics	0.8524	0.3995	0.3324	0.9792
Fuel system	0.9584	0.0	0.0	1.0
Oil system	0.7867	0.3928	0.3773	0.9560
Transmission	0.9530	0.3987	0.2826	0.9906

Table 5.5: Additional metrics of the baseline model regarding class imbalance, *i.e.* the ratio between 'service' and 'no service'. The last two columns regard whether the predictive performance of the model is significantly better than the no information rate at the $\alpha = 0.05$ level.

Target	No service/ service	NIR	Balanced accuracy	P-value (Acc. > NIR)	Significant
Air system	2361/1109	0.6804	0.6356	$< 2.2{\cdot}10^{-16}$	Yes
Brakes	3268/202	0.9418	0.5952	0.5187	No
Electrics	2789/680	0.8040	0.6558	$6.236{\cdot}10^{-14}$	Yes
Fuel system	3326/144	0.9585	0.5	0.5221	No
Oil system	2455/1015	0.7075	0.6667	$< 2.2{\cdot}10^{-16}$	Yes
Transmission	3286/184	0.9470	0.6369	0.0168	Yes

5.2.2 Model Tuning Performance

Based on the results of fitting the baseline models to a selection of target parameters, the oil system and transmission were further investigated by performing model optimisation as described in 4.2.2. The performance when fitting models using "oil system" as target variable is displayed in figure 5.3 and table 5.6, and then using "transmission" as target variable in figure 5.4 and table 5.7. Figures 5.3 and 5.4 gives a graphical overview of model performance across the optimisation parameter grid, showing average model performance as a surface plot. Tables 5.6 and 5.7 summarise the best and median performance across the training grid. The median values were used rather than average values due to the large variation in performance across the training grid potentially skewing the latter.

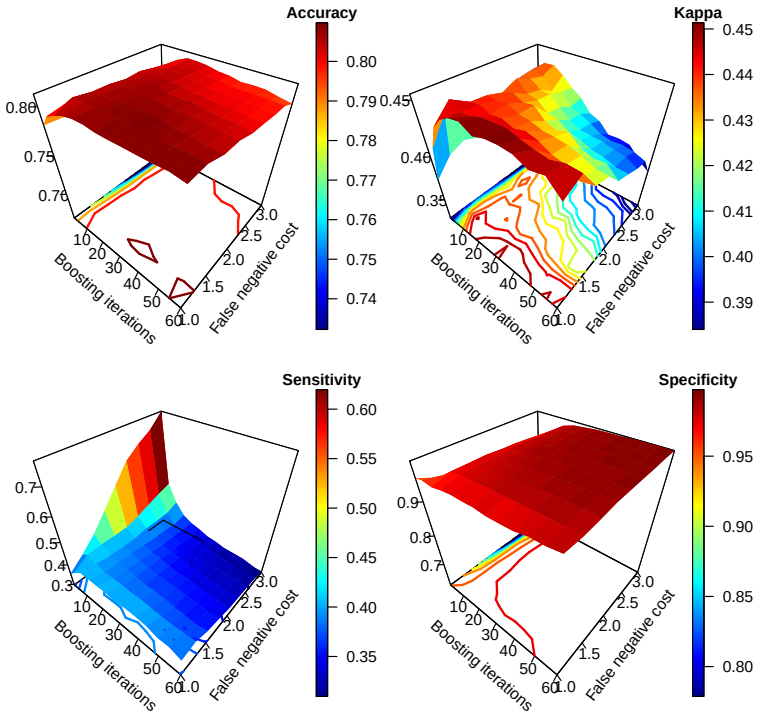


Figure 5.3: Model predictive performance over the training grid with regards to accuracy, kappa value, sensitivity and specificity using "Oil system" as target variable.

The final model fitting, using the model parameter combination which maximised the kappa metric during optimisation, for the two target variables is summarised in table 5.8. In addition to the five metrics used during model optimisation, the difference from the baseline model is also given. In addition to raw performance metrics, table 5.9 gives the 10 most important variables in the best model fits made. The variable importance is given in terms of the percentage of the total number of splits in the model that involves a predictor. A high percentage implies that a variable is often used as a predictor in the model.

Table 5.6: Performance metrics during model optimisation using "oil system" as target variable. The number of boosting iterations and false negative penalisation cost are given together with the corresponding performance metrics, along with the median model performance over the training grid.

Metric	Median value	Maximum value	Best boost	Best weight
Accuracy	0.8044	0.8108	35	1.25
Kappa	0.4296	0.4544	25	1
Sensitivity	0.3717	0.7743	1	3
Specificity	0.9881	0.9976	60	2.75
Balanced Acc.	0.6789	0.7032	1	2.5

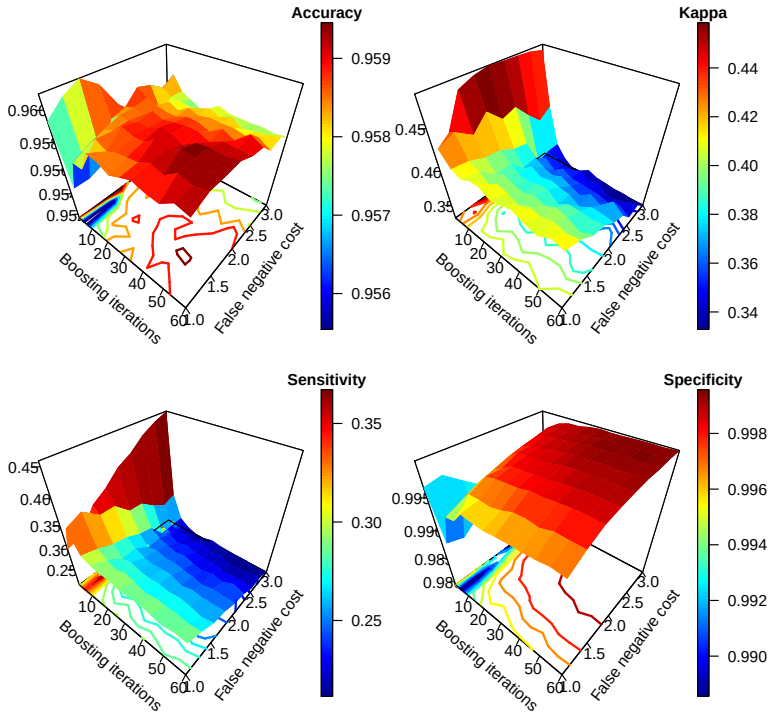


Figure 5.4: Model predictive performance over the training grid with regards to accuracy, kappa value, sensitivity and specificity using "Transmission" as target variable.

Table 5.7: Performance metrics during model optimisation using "transmission" as target variable. The number of boosting iterations and false negative penalisation cost are given together with the corresponding performance metrics, along with the median model performance over the training grid.

Metric	Median value	Maximum value	Best boost	Best weight
Accuracy	0.9586	0.9609	1	1.5
Kappa	0.3847	0.4829	1	2
Sensitivity	0.2565	0.4583	1	3
Specificity	0.9954	0.9996	50	2.75
Balanced Acc.	0.6277	0.7188	1	3

Table 5.8: A summary of the performance of the models fitted by using the best model tuning settings with regards to the kappa metric. The difference compared to the performance of the baseline model fitted without any tuning is also stated.

Metric	Oil system	Difference	Transmission	Difference
Accuracy	0.8051	+0.018	0.9532	+0.002
Kappa	0.4508	+0.056	0.4217	+.023
Sensitivity	0.4241	+0.047	0.3533	+0.071
Specificity	0.9625	+0.007	0.9867	−0.004
Balanced acc.	0.6933	+0.027	0.6700	+0.033
NIR	0.7075	-	0.9470	-

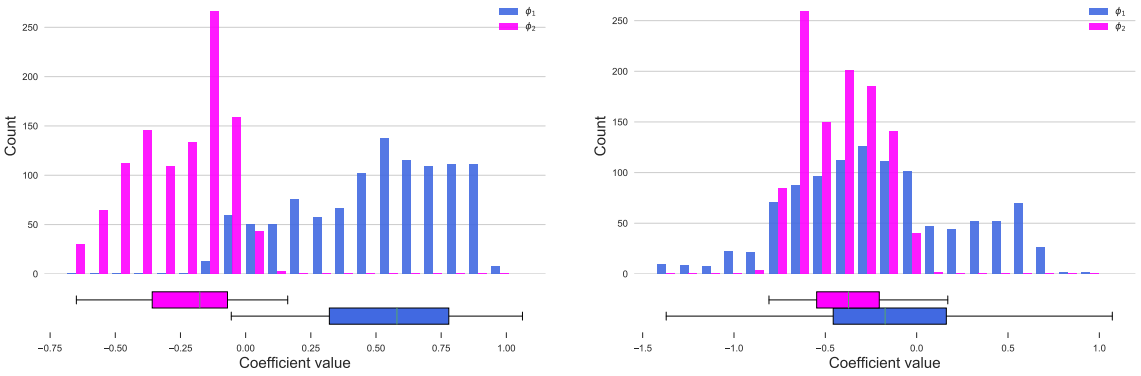
Table 5.9: A summary of the most important predictors when using oil system and transmission as target variables. The variable importance is measured in terms of the percentage of times that a variable is used in the classification model.

Oil system		Transmission	
Variable	% of splits	Variable	% of splits
Engine parts	5.39	Oil system	11.11
Electric system	5.32	Exteriors	5.56
Cooling	4.69	Sealings	5.26
Filters	4.53	Air system	2.78
Sealings	4.53	Battery	2.78
Exhaust system	4.46	Brakes	2.78
Air system	4.22	Combustion chamber	2.78
Exteriors	4.22	Control unit	2.78
Control unit	3.99	Conversion kit	2.78
Fuel system	3.91	Cooling	2.78

5.3 Time Series Modelling

Time series modelling was attempted using two of the logged parameters, and the results of this modelling are reported in this section.

Figures 5.5-5.6 and table 5.10 summarise the AR(2) models fitted to logs of the cumulative engine run time of the trucks. In figure 5.5 and table 5.10 the distribution of the fitted coefficients ϕ_1 and ϕ_2 is presented. Figure 5.6 instead shows a heatmap of the pole placements of the linear system function corresponding to each fitted model.



(a) Distribution of model parameter values for using the original data.

(b) Distribution of model parameter values for using the detrended data.

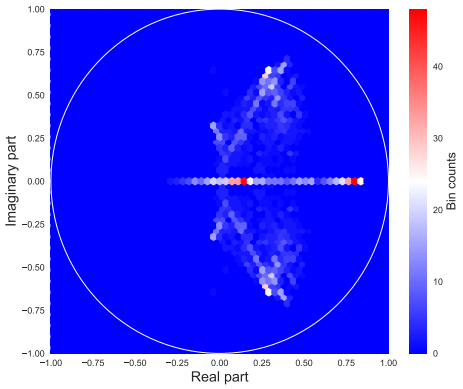
Figure 5.5: Parameter distribution of the parameters fitted to the truck-wise time series data of total engine runtime, obtained by fitting an AR(2) model to the time series of values logged by each truck.

Table 5.10: A summary of the coefficient distributions given in figure 5.5. The metrics are based on a total of 1064 model fits to data series of values coming from different trucks.

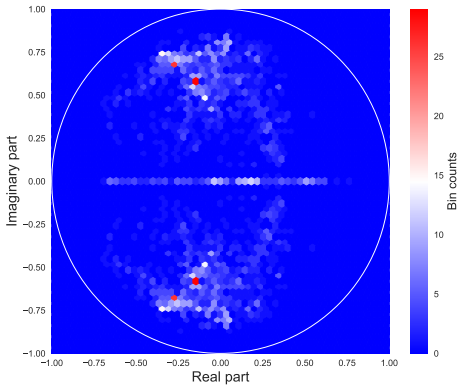
	Original			Detrended		
	Mean	Variance	Min/Max	Mean	Variance	Min/Max
ϕ_1	0.542	0.081	-0.055/1.062	-0.136	0.209	-1.371/1.071
ϕ_2	-0.212	0.031	-0.649/0.161	-0.371	0.039	-0.808/0.170

Table 5.11: A summary of the coefficient distributions given in figure 5.7a. The metrics are based on a total of 4112 model fits to different series of values.

	Mean	Variance	Min/Max
ϕ_1	0.679	0.107	-1.156/1.308
ϕ_2	-0.121	0.025	-0.796/0.673

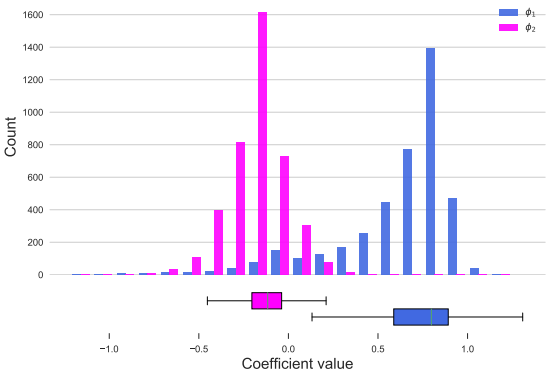


(a) Heatmap of linear system pole locations for models fitted using the raw data.

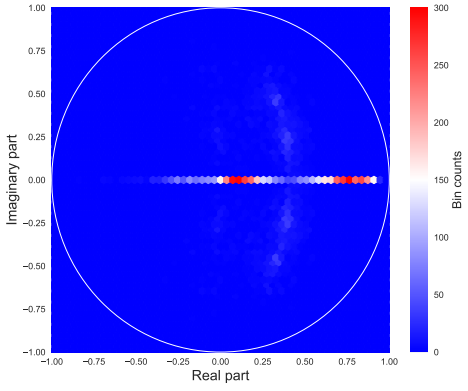


(b) Heatmap of linear system pole locations for models fitted using detrended data.

Figure 5.6: Heatmaps of the pole placements of the linear system representation of the autoregressive models fitted to the truck-wise time series data of total engine runtime. The unit circle is marked to emphasise pole locations.



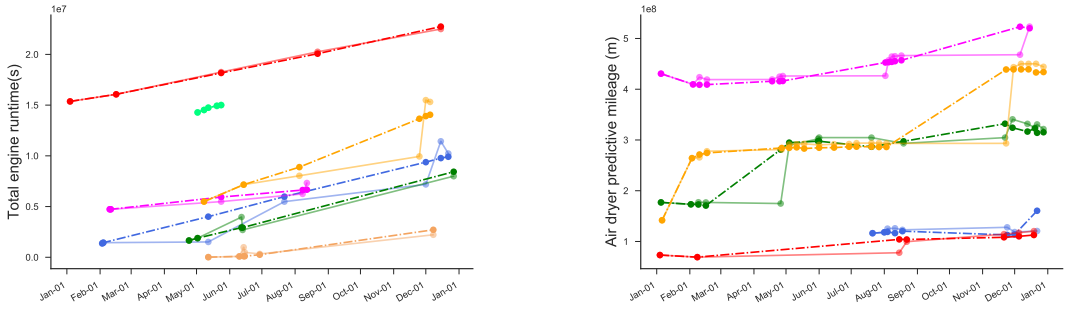
(a) Coefficient distribution of the AR(2) models fitted to the data.



(b) Heatmap of the linear system model pole placements.

Figure 5.7: AR model coefficient distribution, (a), and the corresponding pole locations as a heatmap in (b), when using air dryer predictive mileage as underlying data.

When performing in-sample predictions using the fitted models some of the resulting data series predictions can be seen in figure 5.8. In these plots the original data is shown using solid, opaque lines while the predictions are shown using dashed lines. Note that only the original data series of the engine runtime predictions were used for predictions and not the detrended data.



(a) Data series together with in-sample predictions for the engine runtime data.

(b) Data series together with in-sample predictions for the predictive mileage data.

Figure 5.8: A comparison of the actual data series, plotted as opaque, solid lines, and predictions made by the AR(2) model fitted to the data series for the two parameters modelled.

6

Discussion

In this chapter a number of results from this thesis will be discussed. First of all a thorough discussion of the data exploration carried out will be given. This discussion will touch the prerequisites on the data used for predictive maintenance, and how the data used in this thesis relates to these requirements. This discussion focuses on the results found during the data exploration described in chapter 3. Subsequent sections regard the results obtained from data mining, in 6.2, and service predictions, in 6.3, of the service data. Finally the results of autoregressive modelling of the truck parameter logs is discussed in 6.4.

6.1 Data Quantity and Quality

The field of predictive maintenance is, like machine learning in general, an inherently data-driven science. Because the implementation of a predictive maintenance scheme is in a physical system, creating a model of system degradation and fault diagnostics require that enough data about the system is available. Additionally, this data needs to be relevant to the intended usage of it. As such, not only the *quantity* of the underlying data is of importance, but also the *quality* of it. These requirements were the reason why a thorough process of data exploration was needed, before any more specific analysis of the data provided for this thesis could be made.

The raw data provided for this thesis consisted of just under 30 GB of data. As mentioned in section 3.2 the majority of this data was contained in the truck log data file, with a only a few hundred megabytes of data being in the service data. There were a considerable number of observations available in both datasets, whether in the form of a logged parameter value or in a logged service occasion. However, during data exploration it was discovered that the number of usable observations were much less that the number of observations in the raw data.

Perhaps one of the most troublesome discoveries made was that there was no reliable link between data uploads in the log data and the service data. As such, it was not possible to establish a causal relation between when a parameter value had been logged and any parts that had been serviced. This lack of connection between the datasets provided was the primary reason why they were treated separately.

6.1.1 Log Data Continuity and Sparsity

In the parameter log data the largest loss of usable data stemmed from having to exclude roughly 95% of all data entries simply because they were logs of parameters related to "ranking statistics" of the trucks. As for the remaining data after this exclusion, there were significant differences between the logged parameters, in terms of the number of observations each, which values were logged for each parameter and how many of these logged values were usable. Regarding the number of observations of each parameter there was a clear difference between the type of value logged by a parameter and how many observations were available. The rarest parameters only occurred a handful of times in the entire dataset and, as mentioned in 3.2.2, all these observations were from the same truck and date. In addition to this, the exact value logged by each parameter was obfuscated by the index levels of each parameter. Owing to this, a parameter with for instance five different index levels would in effect only have one fifth of the total number of parameters observations for each parameter-index combination. This significantly reduced the number of usable observations, as not all index level combinations were of interest for each parameter. This was further emphasised by the fact that many parameter-index combinations logged the same type of data for different parameters, or that they were logging constant values with no relation to the condition of the trucks.

The overall consequence of having large amounts of redundant information in the data was that a lot of time and effort had to be put into handling and accessing the data, as explained in section 3.2.2. Moreover, a lot of time was also needed to determine which parts of the log data that were actually of interest, and in turn which parts of the interesting data that were usable for predictive maintenance purposes. In this case, the large amount of data containing unusable information served mostly to make analysis of it more difficult.

Another issue with the parameters logs was the consistency of sampling. When looking at all entries in the data made by each individual truck there was a large difference between each vehicle in terms of both the data readout frequency and total number of entries made by each vehicle. Especially the large time spans with no data samples seen in figure 3.1 indicated continuity problems in the data. For some parameters only some trucks had logged any values, and each such vehicle had only one or two readouts in total. Even the parameters having the most observations in total had issues with the number of data entries per truck. Since one of the goals of this thesis was to find a way of continuously monitoring truck condition, having only at most a few hundred observations in total from a truck, sampled at a few times during a year made any analysis of normal behaviour and detection of performance anomalies infeasible. The practical implications of the infrequent sampling will be further discussed in section 6.4.

As briefly mentioned in 6.1, the infrequent data sampling also raises issues of how to relate the contents of the log data with the contents in the service data. While there did exist a "visit reason" variable in the service data, this variable could not be used during analysis. This was due to that usage of this field had not yet been

implemented in practice during the entire period when data was collected. Thus, it could not be reliably used as an indicator of why a truck was repaired. This fact, combined with the sparse sampling of data in the log data, made it precarious to infer the condition of a truck at the time of service. While it would seem reasonable that only a handful of repairs or service visits are required during a year, having to infer a connection between a service visit and a log data reading two months earlier was dismissed as hopelessly unreliable. This conclusion was the main reason for abandoning the hopes of diagnosing repairs and service of the trucks based on their condition and instead focusing on each set of data separately.

The sparsity of usable data relates both to the quantity, and also to the quality, of the provided data. While collecting large amounts of data is important when seeking to use it for machine learning applications, it is also important to make sure that the data collected is relevant to the intended usage so that data is not unnecessarily collected. In a predictive maintenance framework, it is highly desirable to be able to closely monitor the condition of a system. As such, the data sources need to be carefully chosen so that they are related to areas of interest in the system. Signals and data from these sources must also be possible to be sampled in such a way that the development over time can be accurately followed and related to the condition of the system as a whole.

6.2 Association Rule Mining

As presented in section 4.1, the goal when mining the service data was to investigate whether there were any dependencies or latent structure in the data. Given the rather high number of association rules generated when applying the apriori algorithm to the transaction list made from the service data, as seen in table 5.1, the rules were not evaluated exhaustively in textual form. However, an indication of the kind of rules found was obtained by summarising them as in tables 5.2 and 5.3.

When ordering the mined association rules as in table 5.2 it is apparent that many rules involving structural parts such as members, the grille and the underrun guard stick out. Recall from definition 2.5 that the lift of a rule is a measure of how large the support of an association is compared to when the items involved are independent. A rule with a high lift indicates that the items involved in the rule often occur together. While the support of these rules is quite low, meaning that they appear few times in the transaction data, the high lift makes them stand out in terms of rule importance.The high lift of the rules involving structural parts is thus an indication that there is a tendency for these types of parts to be serviced at the same time.

Looking at the generated rules sorted by their support rather than their lift in table 5.3 one can infer that these rules seems to capture the item distribution of figure 4.1. Comparing the rule table and the item frequency plot it is obvious that the highest support rules all involve the most common items in the transaction data.

The high support of these rules, coupled with the high item frequencies of the items involved, imply that these rules correspond to many of the "routine" types of service that a truck undergoes, such as oil changes, filter changes and the like.

To obtain a better overview of the generated rules one can look at the graphical visualisation of them in figure 5.1. The force directed layout algorithm used to create this graph has the property that it causes the layout of the graph to reflect the structure of the underlying rules. As briefly mentioned in section 5.1 this is done by causing nodes to either attract or repel each other, based on the connections between them. This causes clusters of nodes to form whenever there are many rules associated with an item, and the size of the cluster in turn reflects the number of rules associated with the item. Edges between clusters in turn represent associations between items.

Looking at the most prominent structures in figure 5.1 there seems to be a number of larger clusters present. To further investigate this the OpenOrd force directed layout algorithm was used to create figure 5.2. In this plot there are clearly some distinct clusters present, marked with numbers (1)-(3). Cluster (1) is a collection of the nodes in the upper left corner of figure 5.1. The types of item nodes placed in this cluster are mainly structural ones, such as bumpers, housings, fenders, heat shields and supports. Cluster (2) is a little less homogeneous, than the first cluster. A large number of item nodes in this cluster are related to structural pars such as deflectors, grilles and covers. However, the cluster also contains most of the item nodes related to the electrics of the trucks, such as actuators and electric system components. Relating the contents of cluster (2) to the sparser graph of figure 5.1 is seems like many of the smaller clusters located in the bottom left has been collected into one.

The last and largest cluster, number (3) consists of the nodes in the right part of figure 5.1. The main cluster consists of item nodes such as the fuel- and oil systems, which are the largest nodes in terms of node degree, as well as item nodes such as filters, exhaust system, air system, compressors, pipes and pumps. The smaller clusters surrounding the main cluster are also quite interesting. The bottom one contains item nodes related to the combustion chamber and burner units, with edge connections to rule nodes involving primarily the oil and engine systems. The bottom left subclusters contain the cooling items and is interestingly enough placed on its own. However, considering that it is located on along edges between clusters (2) and (3) this could be an indication that the cooling tends to be serviced at the same time as items in either (2) or (3), although primarily with those in (3). The same goes for the subcluster just above, containing the structural part "stay". Having this node connecting the structural parts in cluster (1) with the components in cluster (3) could also be an indication of a tendency for these types of parts to be replaced simultaneously.

Overall the ability to cluster the association rules as in figures 5.1 and 5.2 does indicate some form of structure in the service data. The constituents of the node clusters found are reasonably similar to what one would expect. Intuitively it makes

sense that large structural parts would be serviced at the same time, or that several types of service to the oil and fuel system would be done at once. This intuition also matches with the rules presented in tables 5.2 and 5.3. To an expert in the field of truck maintenance, such as the mechanics who carry out the actual truck service, these kind of relations are likely no news. It is, however, quite interesting and reassuring to see them turn up during rule mining to serve as an indication that the mining process is finding realistic association rules.

6.3 Service Predictions using C5.0 Classification Trees

The decision to attempt service predictions was based on the structure in data discovered when performing association rule mining. Since there seemed to be some form of structure in the data which could be exploited by a classification algorithm, it was considered feasible to try and distinguish service types from one another.

6.3.1 Model Selection

The process of model selection, in terms of which classification model to use, was briefly touched upon in 4.2.2. The final choice of using the C5.0 classification tree algorithm will be discussed and further motivated here.

Many of the models considered were different types of decision tree models, and this was no coincidence. Given the type of classification problem at hand, with several dummy variables and a few numerical values, using a linear or nonlinear model with coefficients for each variable would be rather hard to interpret. Interpreting such a model could mean considering statements such as service “A” being predicted by 0.5 times service “B” plus 0.33 times service “C” and the likes. Having a model dealing with fractions of binary variables in this way might work well performance wise, but is rather unintuitive in terms of the underlying data. In a practical situation one would likely rather have an all-or-nothing inclusion of each service variable. This is where decision tree models provide a very suitable model structure.

As mentioned in 2.2, classification tree models are essentially a sophisticated series of *if-else* statements used to make a classification based on model inputs. This decision making process is a form of rule based learning, which in turn is a form of expert system learning. In a real life situation, an expert would be the one determining which parts should be serviced, based on current truck condition. By using a rather ’human-like’ decision model such as a classification tree, hopes were that such a model would capture the same type of reasoning that such a human expert would use.

In addition to the model interpretation point of view, model selection was also based on potential model performance. As described in section 4.2.2, a number of different models were initially considered. Most of these models were tree based models, although a support vector machine model was also included. The SVM was included to see if a non-tree, linear classifier would perform substantially better than the tree based classifier. Amongst the tree based models, the level of complexity in each model reflected the performance quite well. The simplest of the four tree models, the ordinary CART-model, fits only one classification tree to the data, pruning the model to a suitable size in the process. Being the simplest model, the performance was also the worst. Random forest, Gradient Boosted trees and C5.0 are all ensemble methods, using either boosting or bagging to increase performance. The increased complexity of using an ensemble of trees rather than just one is obvious, as can be seen in table 4.3. Although increased complexity also brings with it larger computational load, the increase in performance outweighed the computational cost increase in this case, as good predictive performance was an essential end goal. Since the C5.0 algorithm met both the model interpretation criterion, and also performed the best out of all methods tested, it was ultimately selected for further service prediction.

6.3.2 Baseline Performance for Different Targets

Despite C5.0 being found to be the best performing algorithm for the service data, tables 5.4 and 5.5 reveal that there were some potential modelling issues at hand. Looking at the C5.0 model fits to the six parameters used for comparison, it is apparent that the predictive performance on the test data varies depending on which target parameter is used. In table 5.4, the model accuracy seemingly indicates that the model is performing very well on some parameters. When using brakes, fuel system or transmission as targets, around 95% accuracy is achieved, with 75%-85% accuracy on the other targets. However, when looking at the sensitivity and specificity of the model in the same table it is obvious that there is a disparity between the two classes of the target variable. For all target variables, the specificity is high indicating that true negatives, *i.e.* instances of no service, are correctly classified in more than 95% of all cases. At the same time the low sensitivity of at most just under 38% indicates that the ability to predict true cases, *i.e.* service occasions, is much lower. This is exemplified in particular when using "fuel system" as target. In this case the sensitivity of 0.0 and specificity of 1.0 indicates that the model has failed to capture the data, and has simply classified all data as "no service".

For target parameters other than "fuel system", the kappa values show that the model is performing better than what a random classifier would do. This can also be seen by comparing the accuracy in table 5.4 with the no information rate of table 5.5. This shows that the model is performing better than what a classifier assigning only the majority class to all data would do, indicating that the model used is able to capture some relations in the underlying data. However, comparing instead the no information rate to the balanced accuracy of the model shows that model performance is worse compared to a majority class classifier when taking

classwise performance into consideration. For all targets the balanced accuracy is substantially worse than the raw accuracy of the model. Referring to [56], this difference indicates that the raw accuracy is giving an overly optimistic measure of model performance, as well as indicating a potentially biased classifier.

Table 5.5 sheds some light on one of the probable reasons for the target dependent performance seen in table 5.4. Looking at columns 2-3 of table 5.5 it can be seen that the ratio of the two outcome classes varies greatly between target parameters. The no information rate, being the fraction of the data having the majority class label, indicates that there is a severe case of class imbalance for all target parameters. The most balanced cases, using "air system" and "oil system", has a class ratio of roughly 2:1 between no service to service occasions, while using "brakes", "fuel system" or "transmission" has a class ratio of around 20:1. Since classification algorithms tend to perform poorly on unbalanced data, as is the case here, it is obvious that the classification of service occasions has been made more difficult by class imbalance.

6.3.3 Model Tuning and Performance Improvements

Model tuning was used to see if predictive performance of the baseline models discussed in section 6.3.2 could be improved. Only two of the targets used for baseline model fitting were used during tuning, selected by considering model performance on each target. "Oil system" was selected due to both the fact that it had shown the best performance during baseline fitting, and that it was the most class balanced of all targets tested. "Transmission" was selected because it had the best model performance of all the "severe" service types tested. When selecting targets based on a prior performance measure there could be a risk of introducing a bias to the subsequent modelling. In this case, this bias would be an exaggerated relation between the target service and the predictor variables. However, the research question in this thesis was to investigate whether or not the underlying data could be used for diagnosing faults and services at all. Thus, during the modelling process it was considered reasonable to use the most promising service types as targets, in order to see if any data modelling was possible.

The process of model tuning for predicting the presence of service of the oil system is summarised in figure 5.3 and table 5.6. Looking at the overview of model performance over the tuning grid used some inference about model performance can be made. In general, it seems that increasing the number of boosting iterations used in the model increases both accuracy and specificity. However, while specificity is continuously improving with the number of boosting iterations used, the accuracy reaches a maximum when using 35 iterations as can be seen in table 5.6. Sensitivity, on the other hand, shows a decrease with the number of boosting iterations used and is instead greatly enhanced by the introduction of misclassification cost. Unfortunately, the increase in sensitivity is coupled by a corresponding decrease in model specificity, causing overall accuracy to decrease. Since the misclassification penalisation is specifically aimed at increasing sensitivity of the model, it is not sur-

prising that a high penalty increases the number of positively classified oil system service occasions. It is interesting to note that while boosting increases the specificity, and weighting increases the sensitivity of the model, that increasing both at the same times causes sensitivity to decrease to a minimum, while increasing only specificity. Looking at the kappa-metric as a measure of overall model performance, it can be seen in figure 5.3 that there is a ridge of high kappa values along the boosting iteration axis. While increasing the number of boosting iterations is not improving the kappa value further after the maximum at 25 iterations, it is reduced when introducing weighting.

Comparing the results from using "transmission" rather than "oil system" as target some notable differences are revealed. Referring to figure 5.4 and table 5.7, it can be seen that the sensitivity and specificity of the transmission classification model behaves similarly as for the earlier model for the oil system. Sensitivity is once again boosted by the misclassification penalisation, but increasing both the penalty and the number of boosting iterations causes the sensitivity to drop to a minimum while specificity increases. Moreover, model accuracy is improved by increasing both the amount of model boosting and misclassification penalisation, at least for moderate penalties. The accuracy of the oil system model was quite even across the tuning grid, barring when using hefty penalisation and no boosting. For the transmission model, however, there is a distinct accuracy decrease as the penalisation is increased. Overall, as evident in table 5.7, the transmission model shows no increased performance when using boosting, while a large penalisation improves all measured performance metrics. Considering that the class imbalance was very pronounced for the transmission variable, this result makes sense. When boosting, each sub-model trained only sees a fraction of the data. Thus, in the unbalanced dataset it is likely that many of the booster models only see a handful observations of the minority class. These models are then likely to make the same mistake as the fuel system baseline model, only assigning the majority class to all samples. Misclassification penalisation, on the other hand, should in theory help to make the few observations have a larger impact during model fitting, up to a certain limit as discussed in the case of the oil system model. The performance of the two models fitted using the tuning settings which maximised the kappa metric in are summarised in table 5.8. From this table it seems that tuning has indeed had a result in terms of increasing model performance. Comparing these values to those of the baseline models in table 5.4 all metrics except for the the specificity when using transmission as target are improved in the tuned models. The accuracy increase is only marginal, and given the unbalanced data it is in this case not the best performance indicator as discussed at the start of this section. However, the increase in the kappa, sensitivity and balanced accuracy of the models are much more substantial. The increased sensitivity indicates that around 5-7% more cases of a service taking place are correctly classified. Meanwhile, the increase in kappa places the agreement between the fitted model and a perfect classifier as "moderate", rather than "fair" for the baseline model, using the limits specified in [57]. Although, as also stated in [57], these kappa-limits are quite arbitrary, this increase still places the tuned models a step above the baseline model in terms of general agreement between classifications made by the model and the real cases.

An interesting aspect of the final models fitted is to look at the variable importance in the models, *i.e.* which variables that were considered to be the most important predictors. Looking at table 5.9 there are clear differences between the model for prediction oil system service compared to the model for predicting transmission service. For the oil system model, none of the other service types or other variables used stand out as an especially good predictor. Several service types are used in around 4-5% of all splits made by the model. However, comparing these to the association rules in table 5.3 one can see that many of the items occurring in the highest support association rules reoccur in the list of important variables in the classification model. Considering that the association rules could be represented as a rather clustered graph, as in figure 5.1, there could very much be some predictive value in the fact that the rules had such a structure. On the other hand, for the transmission model there is one predictor which sticks out from the rest. Coincidentally, this predictor happens to be the oil system. The exact extent of a service is not clear from the underlying data, and that especially the "oil service" service type was quite broad after the data pre-processing phase. Thus, it is hard to know just from the model just why the presence of oil system service is such a prominent predictor of transmission service. An expert in truck service and repair could perhaps be able to give a hypothesis for why this is the case. Furthermore, both vehicle age and mileage were included as parameters during modelling. However, especially for the oil system service type, it is intriguing that neither of these variables were used as predictors. Since the oil system is a typical part that is regularly checked and serviced, it could perhaps be expected that age and mileage would be good predictors of this kind of service. As for oil system being selected as a predictor of transmission service, the reason for this is not clear from the model. It could be that the classification tree model used does not use this information efficiently, or that these two predictors are simply not as good indicators as one would expect.

Despite the performance improvements obtained after model tuning, the best models obtained still have rather mediocre performance. Nowadays complex models, such as neural network models, have made almost perfect predictive performance the norm in many classification tasks. Although one perhaps shouldn't directly compare different classification tasks, since they are after all very data dependent, close to perfect classification performance should still be expected if the classifier is to be deployed in a live system. Even the best model used in this project had quite poor performance on detecting when service had been carried out on a truck. As such, it would likely not be very suitable for trying to predict if a type of service should be carried out or not in a live setting, such as a workshop.

6.4 Autoregressive Modelling

The point of trying to model the time series of data uploaded from trucks was to see if these data series could be modelled and forecast. However, as already mentioned in 6.1, the lacking quality of the underlying data provided a severe limitation to the

viability of the modelling performed. Despite this, autoregressive modelling of the data series was attempted simply to see if there was any useful information at all to be obtained from the data. In this section focus will be on the actual modelling results, as issues relating to the underling data have already been discussed in 6.1.

Firstly it is worth noting that the model order used for the autoregressive modelling was quite arbitrarily chosen. Due to the low number of data points in each data series, a low model order was required, as using higher order models were deemed to be unsound from a statistical perspective. The uneven sampling of data also meant that the time between two data points was in general be non constant. In an attempt to compensate for this a second order lag term in the autoregressive model was included to could take into consideration larger differences in data over large gaps. No further attempts at verifying whether the model order selected was high enough, or if it was actually too high, was performed. While there are methods of verifying model size, evaluating such statistics from the underlying data was once again considered to be unreliable.

For models fitted using the engine runtime parameter values, it is clear from figures 5.5 and 5.6 that detrending the data makes a large difference in terms of the models fitted. Looking at the coefficients of the fitted models, figure 5.5 makes it clear that very different models are fitted to the raw/detrended data. The models fitted to the original data have a larger difference between the two model coefficients, with ϕ_1 being positive and ϕ_2 being negative. For the detrended data, there is a large variance in the values of ϕ_1, indicating a large variation in the underlying data series. Looking at where the autoregressive model roots are located in figure 5.6, some inference about the type of models fitted can be made. When fitted to the original data, it can be seen in figure 5.6(a) that most roots lie along the real axis, with some complex conjugated root pairs located in the first and fourth quadrants of the unit circle. As described in [50], this type of pole placements implies that the autocorrelation of the data most often behaves as a mixture of two exponentially decaying terms. For the models where the roots are complex the autocorrelation behaves as a damped sinus function, as exemplified by figure 2.3(a). Models having a monotonically decaying dependence on previous values as seems natural when considering what the underlying data looks like in figure 4.2. In this figure most data series follow a linearly increasing trend, with some deviation at times. Thus, it makes sense that old values have little impact on newer ones. Looking instead the pole placements of the models fitted to the detrended data, seen in figure 5.6(b), there is a noticeable difference compared to the original data. For the detrended data most poles are complex, and only a few lie along the real axis. Thus, for a majority of the detrended data series the autocorrelation function behaves in an oscillatory way. Looking at the underlying data series in figure 4.2(b) it can be seen that the detrended values tend to oscillate between positive and negative. In this way it makes sense that a larger number of models are capturing this kind of oscillatory behaviour than those fitted to the original data. Despite the lack of data, this implies the the autoregressive models are at least capturing some of the information carried in the underlying data series.

The data associated with the air dryer predictive mileage parameter had much less of a trend than the engine runtime parameter, as seen in figure 4.3. As such, no detrending was done and only the original data was modelled. Looking first at the coefficient distribution for the fitted models in figure 5.7(a) it is clear that the two model coefficients ϕ_1 and ϕ_2 are more distinct than for the engine runtime models. Once again a positive ϕ_1 and negative ϕ_2 imply a slightly oscillating dependence on previous values, although weaker than in the previous models as seen by comparing tables 5.10 and 5.11. Looking at the pole placements in figure 5.7(b) one can see that a majority of the model poles are real valued, indicating an exponentially decaying autocorrelation function for the data. Once again this is not a big surprise, considering the sample data in figure 4.3. As with the engine runtime data series the lack of data points gives few quick variations in data between observations. As such it makes sense that models fitted to the data have little dependence on older values.

Since there was no specific sampling rate for the data, forecasting using the fitted models was not performed. While nothing stops generating new data using the fitted model coefficients, there was no way of knowing at which points in time these predicted values should be made. Thus, interpreting these predicted values as future data values was not reasonable. Instead of forecasting, in-sample predictions were used to evaluate how well the AR models captured the data series. Since the underlying data varied a lot between different models no overall performance measure was used. Instead, a few data series were plotted to visually evaluate the model fit as in figure 5.8. As can be sen in this figure, the models are able to quite accurately capture the data with even sampling and linear trend. However, as can especially be seen in 5.8(b) problems arise when data points are located in small groups separated by large gaps. In these cases, the models tend to fall behind between groups of data points, failing to react to the change in parameter value between different groups. Whether this issue could be solved by, for example, using higher order models or not was not investigated during this thesis. However, trying to fix these kind of prediction issues by tailoring the model would only be to address a symptom of the underlying cause that is the lack of data.

Comparing the models fitted to the different parameters it is not surprising that they differ quite a bit. Since the parameters are logging very different types of information, the resulting data series should naturally have different structures. As such it is only natural that different models, in terms of the fitted model coefficients, are required to capture the two types of data series. Moreover, a large degree of model variation between trucks should also be expected, as the trucks are used in very different ways, are of different ages etc. Therefore one should not expect a single “universal” model to be a viable way of modelling the type of data provided, but that a individual model needs to be fitted using the data generated by each truck. As mentioned several times already, it is in this case vital that the amount of data provided from each truck is sufficient in terms of sampling rate and relevancy for truck condition. The most important aspect when making a model to predict these kinds of parameter logs is that there are enough samples, and that these are made continuously. Frequently sampled data allows for both a more accurate model, as well as a more up-to-date model. Compared to many of smaller, isolated systems

where predictive maintenance has been tested, such as those in studies presented in section 1.3, it is not reasonable to be able to continuously monitor all subsystems of a large technical systems in the same way. However, having some systems on a truck related to fault monitoring, and tracking these systems on a frequent basis would be a good start for implementing a predictive maintenance framework in trucks as well.

Overall, owing to the small amount of data that each autoregressive model is fit to the results obtained from the time series modelling should be interpreted with caution. Given the low statistical support for the model fitting process, placing too much weight on the results is ill advised. It is very well possible that more frequently sampled data over longer times could, as discussed in the previous paragraph, could have a large impact on modelling results. As such, the results and conclusions drawn from this work are perhaps better considered to be a start of a data requirement evaluation for future implementations of predictive maintenance in trucks.

7

Conclusions and Outlook

Following an initial literature study of the current state of the field of predictive maintenance, a thorough exploration of the data provided by AB Volvo for this project was carried out. During this data exploration phase it was discovered that the structure of the provided data would not directly allow entries in one dataset to be connected to entries in the other. This resulted in each dataset being subjected to different analytic approaches.

The first part of the data, related to repair and service made to trucks, was analysed using unsupervised data mining methods, as well as supervised classification models for detecting faults. The results and implications of these, as discussed in sections 6.2 and 6.3 can be summarised as

- There is an underlying structure in what types of repairs are carried out at the same time on a truck. Although not groundbreaking, the structure discovered during this project correspond well with service relations that could be intuitively expected.

- Using a classification tree model it is to some degree possible to determine if a certain type of service has been carried out on a truck based on other types of repairs. However, the accuracy of this classification is dependent on which fault is being predicted, and is overall too low to be reliable in a practical application. This is especially true for the more rare types of service, which unfortunately were those that are also rather severe.

The latter part of this project involved modelling and predicting the time series of logged truck parameters. It was discovered that the data supplied was insufficient for the purpose of creating an accurate model of truck condition and usage. Based on the discussion given in section 6.1, these reasons are summarised in the following points.

- *Data sparsity and continuity*, the number of data points per truck was inadequate and unevenly sampled, making modelling of the data series falter. Due to this, care should also be taken when interpreting and drawing conclusion from these models, as the underlying statistics are very rough. For the sake of modelling it would have been better to have fewer but more frequently sampled parameters available, rather than many sparsely sampled ones.

- *Data redundancy*, although a large amount of raw data was provided, the amount of usable data was much less having had to remove, for the intended usage, unusable parts of the data. A large part of the data contained duplicate or redundant information.

- *Data relevancy*, large parts of the data were irrelevant for the purpose of predictive maintenance. However, given that the data was not collected with the intention of using it for predictive maintenance, this should perhaps not be expected.

An overall conclusion from this thesis is that a lot of work is needed with regards to what data is collected before a reliable predictive maintenance framework can be established. The project work has shed light on the importance of having high quality and purposeful data available when burrowing into data driven analytics.

7.1 Future Outlook

This thesis has mainly been an exploratory endeavour, seeking to find new usage of data already collected by AB Volvo. As such, the goal has been to provide insight into the data provided and to investigate whether this data, which is already being collected anyway, can be used in new applications of interest for Volvo. Although the conclusions from using this particular data for the application of predictive maintenance have been rather discouraging, it has still given rise to some possible proposals for further work in the field of predictive maintenance on trucks.

A reasonable course of action based on the results of this thesis would be a project to identify and acquire the kind of data required for an implementation of predictive maintenance in the Volvo truck fleet. As discussed in section 6.1, the success of a predictive maintenance framework is hinged on the quality of the underlying data. Based on the findings in this thesis, it would likely be more productive to make sure that the data used is suitable for use in a predictive maintenance framework. Thus, a thorough data requirement analysis would be a natural first step in assuring data quality and relevance before further work using such data is undertaken.

The issues related to the data quality, which were encountered during this project, could also be taken as a sign that some form of data sanitation project is relevant. Storing vast amounts of data which are never used once collected is wasteful, and contributes no new information to a data driven analytics framework. The data provided for this project likely contain relevant and necessary information within the original domain in which it is used. However, within an entirely different context, such as predictive maintenance, it is apparent that there is a lot of redundant information that is no longer relevant. Thus, a relevant project would be to evaluate what data is collected from trucks, why this data is collected and how it is used later. Such an analysis should result in a better insight in both what data is already available, but also what data is possible to use.

Bibliography

[1] R. Kothamasu, S. H. Huang, and W. H. VerDuin. System health monitoring and prognostics — a review of current paradigms and practices. *The International Journal of Advanced Manufacturing Technology*, 28(9):1012–1024, 2006.

[2] Ying Peng, Ming Dong, and Ming Jian Zuo. Current status of machine prognostics in condition-based maintenance: a review. *The International Journal of Advanced Manufacturing Technology*, 50(1):297–313, 2010.

[3] Elsayed A. Elsayed. *Reliability engineering*. J. Wiley, Hoboken, N.J, 2nd;2. aufl.;2; edition, 2012.

[4] Jong-Ho Shin and Hong-Bae Jun. On condition based maintenance policy. *Journal of Computational Design and Engineering*, 2(2):119–127, 2015.

[5] L. Zhang, X. Li, and J. Yu. A review of fault prognostics in condition based maintenance. *Proc. of SPIE*, 6357:635–752, nov 2006.

[6] A Starr, R Willetts, P Hannah, W Hu, D Banjevic, and AKS Jardine. Data fusion applications in intelligent condition monitoring. In *IFRIM Conference*, pages 6–8, 2002.

[7] Andrew K.S. Jardine, Daming Lin, and Dragan Banjevic. A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing*, 20(7):1483 – 1510, 2006.

[8] Otilia Elena Dragomir, Rafael Gouriveau, Florin Dragomir, Eugenia Minca, and Noureddine Zerhouni. Review of prognostic problem in condition-based maintenance. In *Control Conference (ECC), 2009 European*, pages 1587–1592. IEEE, 2009.

[9] Phil Blunsom. Hidden markov models. *Lecture notes, August*, 15:18–19, 2004.

[10] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[11] C Okoh, Rajkumar Roy, Jorn Mehnen, and L Redding. Overview of remaining useful life prediction techniques in through-life engineering services. *Procedia CIRP*, 16:158–163, 2014.

[12] Chiman Kwan, Xiaodong Zhang, Roger Xu, and Leonard Haynes. A novel approach to fault diagnostics and prognostics. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 1, pages 604–609. IEEE, 2003.

[13] Ratna Babu Chinnam and Pundarikaksha Baruah. Autonomous diagnostics and prognostics through competitive learning driven hmm-based clustering. In *Neural Networks, 2003. Proceedings of the International Joint Conference on*, volume 4, pages 2466–2471. IEEE, 2003.

[14] Faith Camci. *Process monitoring, diagnostics and prognostics using support vector machines and hidden Markov models*. PhD thesis, Wayne State University, Detroit, 2005.

[15] Ming Dong and David He. A segmental hidden semi-markov model (hsmm)-based diagnostics and prognostics framework and methodology. *Mechanical Systems and Signal Processing*, 21(5):2248 – 2266, 2007.

[16] Zhigang Tian. An artificial neural network method for remaining useful life prediction of equipment subject to condition monitoring. *Journal of Intelligent Manufacturing*, 23(2):227–237, 2012.

[17] Nagi Gebraeel, Mark Lawley, Richard Liu, and Vijay Parmeshwaran. Residual life predictions from vibration-based degradation signals: a neural network approach. *IEEE Transactions on industrial electronics*, 51(3):694–700, 2004.

[18] Nagi Z. Gebraeel and Mark A. Lawley. A neural network degradation model for computing and updating residual life distributions. *IEEE Transactions on Automation Science and Engineering*, 5(1):154–163, 2008.

[19] Alice E Smith, David W Coit, and Yun-Chia Liang. Neural network models to anticipate failures of airport ground transportation vehicle doors. *IEEE transactions on automation science and engineering*, 7(1):183–188, 2010.

[20] Gang Yu, Hai Qiu, Dragan Djurdjanovic, and Jay Lee. Feature signature prediction of a boring process using neural network modeling with confidence bounds. *The International Journal of Advanced Manufacturing Technology*, 30(7):614–621, 2006.

[21] T. V. Thomaidis and S. Pistikopoulos. Criticality analysis of process systems. In *Reliability and Maintainability, 2004 Annual Symposium - RAMS*, pages 451–458. IEEE, 2004.

[22] Jihong Yan, Muammer Koc, and Jay Lee. A prognostic algorithm for machine performance assessment and its application. *Production Planning & Control*,

15(8):796–801, 2004.

[23] KB Goode, J Moore, and BJ Roylance. Plant machinery working life prediction method utilizing reliability and condition-monitoring data. *Proceedings of the Institution of Mechanical Engineers, Part E: Journal of Process Mechanical Engineering*, 214(2):109–122, 2000.

[24] Alexandre Trilla and Pau Gratacòs. Maintenance of bogie components through vibration inspection with intelligent wireless sensors: A case study on axle-boxes and wheel-sets using the empirical mode decomposition technique. *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit*, 230(5):1408–1414, 2016.

[25] Ashok Prajapati, James Bechtel, and Subramaniam Ganesan. Condition based maintenance: a survey. *Journal of Quality in Maintenance Engineering*, 18(4):384–400, 2012.

[26] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *Acm sigmod record*, volume 22, pages 207–216. ACM, 1993.

[27] Christian Borgelt. Efficient implementations of apriori and eclat. In *FIMI'03: Proceedings of the IEEE ICDM workshop on frequent itemset mining implementations*, 2003.

[28] Aida Nordman. Lecture notes in TNM033 - Data Mining. `http://staffwww.itn.liu.se/~aidvi/courses/06/dm/lectures/lec7.pdf`, October 2011.

[29] Max Bramer. *Principles of Data Mining*. Springer London, London, 3rd 2016.;3rd 2016; edition, 2016.

[30] Rakesh Agrawal and Ramakrishnan et al Srikant. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499, 1994.

[31] Mohammed J. Zaki, Srinivasan Parthasarathy, Mitsunori Ogihara, and Wei et al Li. New algorithms for fast discovery of association rules. In *KDD*, volume 97, pages 283–286, 1997.

[32] Jiawei Han, Jian Pei, Yiwen Yin, and Runying Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data mining and knowledge discovery*, 8(1):53–87, 2004.

[33] Christian Borgelt. An implementation of the fp-growth algorithm. In *Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations*, pages 1–5. ACM, 2005.

[34] Chengqi Zhang and Shichao Zhang. *Association rule mining: models and algorithms*, volume 2307.;2307. Lecture notes in artificial intelligence;. Springer,

New York;Berlin;, 1 edition, 2002;2006;.

[35] Max Kuhn and Kjell Johnson. *Applied Predictive Modeling*. Springer New York, New York, NY, 2013.

[36] J. Ross Quinlan. {CHAPTER} 4 - pruning decision trees. In J. Ross Quinlan, editor, *C4.5*, pages 35 – 43. Morgan Kaufmann, San Francisco (CA), 1993.

[37] Xindong Wu, Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J. McLachlan, Angus Ng, Bing Liu, Philip S. Yu, Zhi-Hua Zhou, Michael Steinbach, David J. Hand, and Dan Steinberg. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37, 2008.

[38] C. E. Shannon. A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5(1):3–55, January 2001.

[39] J. Ross Quinlan. {CHAPTER} 2 - constructing decision trees. In J. Ross Quinlan, editor, *C4.5*, pages 17 – 26. Morgan Kaufmann, San Francisco (CA), 1993.

[40] John Mingers. An empirical comparison of selection measures for decision-tree induction. *Machine learning*, 3(4):319–342, 1989.

[41] Thomas G. Dietterich. Machine-learning research: Four current directions. *AI Magazine*, 18(4):97, 1997.

[42] Robert E. Schapire and Yoav Freund. *Adaptive Computation and Machine Learning: Adaptive Computation and Machine Learning : Boosting - Foundations and Algorithms*. MIT Press, 2012.

[43] Rulequest Research. C5.0: An informal tutorial. `https://www.rulequest.com/see5-unix.html#BOOSTING`, 2017. Accessed: 2017-05-05.

[44] Robert H. Shumway and David S. Stoffer. *Time series analysis and its applications: with R examples*. Springer, New York, 2nd [updat];second; edition, 2006.

[45] Umberto Triacca. Lecture notes in time series econometrics. `http://www.phdeconomics.sssup.it/documents/Lesson4.pdf`.

[46] G.E.P. Box, G.M. Jenkins, G.C. Reinsel, and G.M. Ljung. *Time Series Analysis: Forecasting and Control*. Wiley Series in Probability and Statistics. Wiley, 2015.

[47] Ali N Akansu and Mustafa U Torun. Toeplitz approximation to empirical correlation matrix of asset returns: A signal processing perspective. *IEEE Journal of Selected Topics in Signal Processing*, 6(4):319–326, 2012.

[48] Peter J. Brockwell and Richard A. Davis. *Introduction to time series and forecasting*. Springer, New York, 2nd edition, 2002.

[49] James V. Candy. *Model-based signal processing*. Wiley, Hoboken, N.J, 1 edition, 2006.

[50] Douglas C. Montgomery, Cheryl L. Jennings, and Murat Kulahci. *Introduction to time series analysis and forecasting*. Wiley, Hoboken, New Jersey, 2nd edition, 2015.

[51] Michael Hahsler, Christian Buchta, Bettina Gruen, and Kurt Hornik. *arules: Mining Association Rules and Frequent Itemsets*, 2017. R package version 1.5-2.

[52] Michael Hahsler and Sudheer Chelluboina. Visualizing association rules: Introduction to the r-extension package arulesviz. *R project module*, pages 223–238, 2011.

[53] Daniel A. Keim, Mike Sips, and Michael Ankerst. 43 - visual data-mining techniques. In Charles D. Hansen and Chris R. Johnson, editors, *Visualization Handbook*, pages 831 – 843. Butterworth-Heinemann, Burlington, 2005.

[54] Roberto Tamassia. *Handbook of graph drawing and visualization*. Taylor & Francis, CRC Press, Boca Raton, 2014.

[55] Gephi. Gephi tutorials layout. `https://gephi.org/tutorials/gephi-tutorial-layouts.pdf`, 2011.

[56] Kay Henning Brodersen, Cheng Soon Ong, Klaas Enno Stephan, and Joachim M Buhmann. The balanced accuracy and its posterior distribution. In *Pattern recognition (ICPR), 2010 20th international conference on*, pages 3121–3124. IEEE, 2010.

[57] J Richard Landis and Gary G Koch. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174, 1977.

[58] Tom Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.

[59] Alan Agresti. *Categorical data analysis*, volume 792. Wiley, Hoboken, NJ, 3rd edition, 2013;2014;.

[60] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960.

[61] Mary L. McHugh. Interrater reliability: the kappa statistic. *Biochemia medica*, 22(3):276, 2012.

A

Evaluation of Classifier Performance

This appendix gives a brief review of metrics used to evaluate the performance of a binary classifier. The metrics regarded here are the accuracy, kappa statistic, sensitivity, specificity and balanced accuracy of the classifier. Only a brief review of each of these will be given. For a more thorough review of these metrics, as well as other metrics used for performance evaluation, the reader is referred to the more comprehensive introduction given in for instance [58, 35, 59].

Table A.1: Confusion matrix, also known as a contingency table, for a two-class classifier with assumed outcomes '0' (negative) and '1' (positive). TN, FN, FP and TP each represent the number of observations present in each quarter of the matrix.

		Reference value: 0	Reference value: 1
Prediction	0	True negative (TN)	False negative (FN)
	1	False positive (FP)	True positive (TP)

A practical way of presenting the results of applying a classification algorithm to labelled data is in the form of a confusion matrix, as can be seen table A.1. In such a representation the columns represent the actual class labels of the data, while the rows represent the predicted classes. Assuming (arbitrarily) that the class labels are '0' and '1', and that these are considered 'negative' and 'positive', respectively. In this setting, the diagonal elements of the confusion matrix, the true positive (TP) and true negative(TN) rates, are the observations which are correctly classified by the algorithm. The false negatives (FN) are positive observations which have been wrongly labelled as negative by the algorithm. Contrary, the false positives (FP) are the negative observations which have been incorrectly classified as positive. Using this notion, a number of metrics regarding the predictive performance of the model can derived [58].

The *accuracy* of the model is simply the fraction of correct classifications made

overall, that is

$$\text{Accuracy} = \frac{TN + TP}{TN + FN + FP + TP} = \frac{\text{Correct predictions}}{\text{Total number of observations}}. \tag{A.1}$$

Naturally, a high accuracy implies that the classifier is able to correctly predict a large number of observations. However, it says nothing about the predictive performance on the individual classes, nor if some type of misclassification is considered more severe than another [35]. To evaluate the performance on each class the sensitivity and specificity metrics can be used instead.

The *sensitivity*, also known as *true positive rate* or *recall* of the model, gives a measure of how many of the positive observations that are correctly classified;

$$\text{Sensitivity} = \frac{TP}{TP + FN} = \frac{\text{Correctly predicted positive observations}}{\text{Total number of positive observations}}. \tag{A.2}$$

A high sensitivity thus implies that the model successfully classifies a large number of 'positive' observations.

Contrary to sensitivity the *specificity*, or *true negative rate*, of the model gives a measure of the fraction of correctly classified 'negative' observations;

$$\frac{TN}{TN + FP} = \frac{\text{Correctly predicted negative observations}}{\text{Total number of negative observations}}. \tag{A.3}$$

Similarly to sensitivity a high specificity of a model implies that many negative observations are correctly classified as such.

The predictive accuracy of equation (A.1) gives a representative measure of model performance when the classes are balanced, *i.e.* when there are about as many observations of each class in the data. However, when this is not the case then the ordinary accuracy can give an overly optimistic measure of model performance. In such situations an alternative accuracy measure known as the *balanced accuracy* can be a better way of measuring predictive accuracy, and it is defined through

$$\textit{Balanced accuracy} = \frac{1}{2} \cdot \left(\frac{TP}{TP + FN} + \frac{TN}{TN + FP} \right) = \frac{1}{2}(\text{sensitivity} + \text{specificity}). \tag{A.4}$$

The balanced accuracy considers the average performance of the predictions on each class. When performance is similar on each class the balanced accuracy is reduced to the ordinary accuracy, while it will be lower whenever the classifier performs worse on any of the classes [56].

Related to imbalanced classes is the *no information rate* (NIR), of the data. As opposed to the balanced accuracy this metric is related to the data rather than the model used for predictions. The no information is defined as the accuracy achieved by randomly assigning classes when predicting. It is thus the accuracy which is expected 'by chance' [35]. For a two-class classification problem the no information rate is taken to be the fraction of the data belonging to the majority class. Any

predictive model fitted to the data should have an overall accuracy in excess of the no information rate if the model is to be of any value [35].

Another way of evaluating model performance compared to that of random guessing is by using the *Cohen's kappa* statistic, denoted by κ. This statistic evaluates the agreement between two independent classifications of observations, taking into consideration the agreement expected by chance [35]. In the case of a binary classification the class labels and the predictions made by the classifier are taken as these two parties. A κ value of 0 implies total disagreement between observations and predictions, that is the model performs no better than random guessing. On the other hand, a κ value of 1 implies that perfect agreement between observations and predictions even after correcting for chance [59]. The kappa statistic is defined [60] as

$$\kappa = \frac{p_o - p_c}{1 - p_c}, \tag{A.5}$$

where p_o is the observed accuracy of the model and p_c is the accuracy expected when predicting random classes. For a description of how the expected accuracy p_c can be estimated the reader is referred to [61].