Applications of time-to-event data analysis in root cause analysis of medical imaging systems

Xia, H.Q.

Award date:
2019

EINDHOVEN UNIVERSITY OF TECHNOLOGY & PHILIPS

Applications of Time-to-Event Data Analysis in Root Cause Analysis of Medical Imaging Systems

Master Thesis (1BM96)

Han qi Xia (0896648)
h.q.xia@student.tue.nl

Eindhoven University of Technology
Department Industrial Engineering & Innovation Sciences
Information Systems capacity group

Philips
Data Science Department

In partial fulfilment of the requirements for the degree of

Master of Science in
Operations, Management & Logistics

Classification:

public

Supervisors TU/e:
dr. Yingqian Zhang
dr. Alp Akçay
dr. Remco Dijkman

Supervisors Philips:
dr. Jan Korst
dr. Verus Pronk

October 06, 2019

Abstract

Philips is a manufacturer of medical imaging systems. In order to optimize the maintenance of these systems, Philips equips their systems with networking software to facilitate the collection of operation and maintenance data. Philips uses this data to power numerous predictive maintenance models, which have proven themselves very powerful tools for supporting the maintenance of specific systems and parts.

Currently, there is an interest in the applicability of survival analysis to develop general methods that will support the maintenance of all system types and parts, using data that is available for their entire product line. A substantial portion of maintenance cases recorded are found to be corrective. The medical domain is unforgiving for downtime, with each additional hour of downtime bringing substantial costs with it. Service engineers strive to fix issues on their first visit, while only a few parts can be brought along.

An in-depth analysis of the data relevant to medical imaging system maintenance currently gathered was performed in order to identify possible solutions. A literature review was conducted to identify and evaluate best practices and standards in survival analysis, root-cause analysis and multi-component system maintenance. Both non-parametric survival analysis and standard machine learning methods were analysed in their applicability for covariate approximation and root-cause analysis.

From the data currently gathered, maintenance data fits the bill for 'universally available data'. Service engineers are already supported by powerful predictive models that can predict imminent failure with great accuracy. Many of these models are reliant on usage data, daily logs or performance tests. This data is not available for all system types and generally focus on one component within the system. Random forest models, were found to be more capable of approximating the usage intensity of systems, using maintenance data, than survival analysis based models.

A root-cause failure prediction algorithm was conceptualized and evaluated. This algorithm returns a list of parts for a given maintenance call, ranked on their probability of being the cause of the failure. Cox Proportional Hazards regression models are fitted to individual parts. The database at Philips is queried in order to obtain the current parts in the system and their age. Several predictors are derived from these models and used as features for a blender model in a stacking ensemble. In several case studies regarding several fault modes, multiple configurations of the algorithm were found to outperform a simple baseline to differing degrees.

Keywords: *time-to-event data analysis, survival analysis, recurrent event data analysis, non-parametric, Kaplan-Meier, product limit estimator, Nelson-Aalen, Cox proportional hazards regression, multi-component system maintenance, machine learning, ensemble stacking, root-cause analysis*

List of Abbreviations

PM	preventive maintenance
CM	corrective maintenance
FCO	field change order
R(t)	reliability
F(t)	cumulative distribution function
f(t)	failure probability density function
h(t)	hazard rate
KME	Kaplan-Meier estimate
CoxPH	Cox proportional hazards
REDA	recurrent event data analysis
RCA	root cause analysis
RCF	root cause failure
Logit	logistic regression
RF	random forest
SMOTE	synthetic minority oversampling technique
SVM	support vector machine
ADA Boost	adaptive boosting
XGBoost	extreme gradient boosting
NN	neural network

Philips/medical imaging systems related abbreviations

MIS	medical imaging system
ISDA	imaging system data analytics
ETL	extract, transform & load
MR	magnetic resonance
CT	computer tomography
IXR	interventional X-ray

Executive Summary

This thesis project was conducted in collaboration with the data science department of Philips Research in Eindhoven.

0.1 Research goal

Currently, there is an interest in the applicability of survival analysis to develop general methods that will support the maintenance of systems and parts of any type. Thus, focus was placed on universally available data. Initial data exploration revealed that a substantial portion of maintenance calls are corrective. Service engineers strive towards 'first-time right', in order to minimize the cost of downtime. These points led to the goal of investigating the applicability of survival analysis in root-cause analysis.

0.2 Methodology

A proof of concept implementation of a root-cause failure prediction algorithm was implemented in the R statistical programming language. The software tool consists of two stand-alone modules. Survival analysis is applied by the first module, called the survival analysis tool henceforth, allowing researchers and engineers to use the tool for separate purposes outside of root cause analysis, such as visualisation of data.

The survival analysis tool applies either the Kaplan-Meier estimate or Cox proportional hazards regression to draw survival curves for individual parts. A supporting model was trained to allow the prediction of system usage intensities using the maintenance calls recorded. The second module applies the root-cause failure prediction algorithm, using the output of the survival analysis tool and data from the ISDA database. Simple predictors are extracted from the fitted Cox models. These predictors are used as features for a meta-learner model, in a stacking ensemble as shown in Figure 0.1.

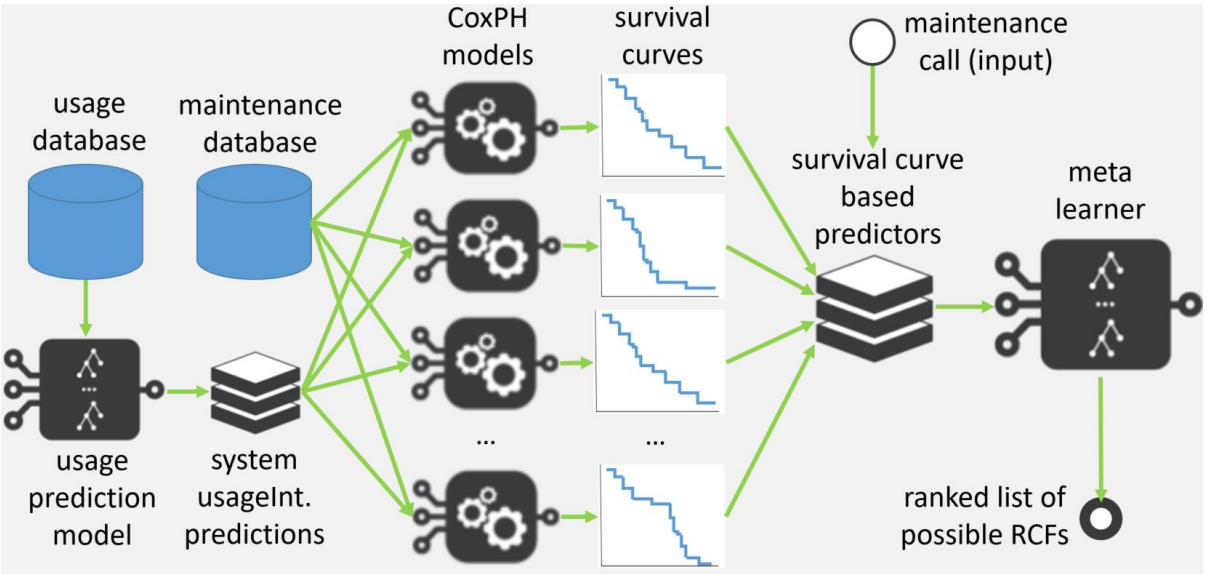


Figure 0.1 – Multi-level stacking ensemble framework

The models were scored on the number of parts that it ranks higher (predicts to have a higher probability to be the root-cause) than the actual root-cause. This can be translated into the number of parts a service engineer has to check before diagnosing the right part, if relying purely on this model. Models are benchmarked against a simple method where parts were ranked solely on their historical frequency in the database.

0.3 Findings and conclusion

Experimentation was limited to a handful of subsystems within a single system type to allow for in-depth analysis. Only generalizable methods and universally available data were used to meet the original objective. Two different multiclass classifier meta-learner models were tested for the C-arm stand subsystem. The results are shown in Table 0.1. The XGBoost model was found to slightly outperform the random forest model and the benchmark.

Table 0.1 – C-arm stand multiclass classifier performance metrics

Model	**Random Forest**	**XGBoost**	**Benchmark**
Ranking score	4.973	4.763	6.090
Proportion not in top 2	0.667	0.603	0.706

Assuming a failure distribution similar to the historical data, the difference in ranking score could very roughly be translated to: '*A service engineer diagnosing failures using the root-cause prediction algorithm would need to check ±1.3 parts less on average than a service engineer checking parts in the order of their historic frequency, before successful diagnosis.'*

Table 0.2 below present the results of an experiment involving 50 parts of a CT system, in a situation of high data availability.

Table 0.2 – Results 'Brilliance CT 64 CHANNEL'

Survival method	**Meta-learner**	**algorithm rankScore**	**benchmark rankScore**	**algorithm % not in top 5**	**benchmark % not in top 5**
CoxPH	RF (mtry=16)	15.49326	19.09723	71.12285	79.65654
CoxPH	XGBoost	14.09194	19.09723	65.65390	79.65654
KME	RF (mtry=16)	16.35641	19.09723	74.55746	79.65654
KME	XGBoost	14.88560	19.09723	69.51123	79.65654
CoxPH	Homogeneous One-vs-all XGB	14.35890	19.09723	67.62915	79.65654

As data availability increases (both in terms of covariates and sample size), greater performance will be achieved using CoxPH regression instead of the KME. As data availability and problem complexity increases, XGBoost gradually outperforms the RF algorithm as the meta-learner, to an increasing degree. With regards to scalability for complex problems (>50

classes), boosted logit trees using a One-Vs-All approach are the best option for the meta-learner, with little loss in predictive power, compared to multiclass XGBoost models.

Performance of different archetypes of meta-learners were evaluated for the 'UI modules' subsystem using the error rate, which is calculated as the proportion of mislabelled instances. An error rate of (0.615) was observed for the benchmark. Multiclass classifiers (0.539) were outperformed by a cascade of binary classifiers using a One-Vs-All approach (0.493).

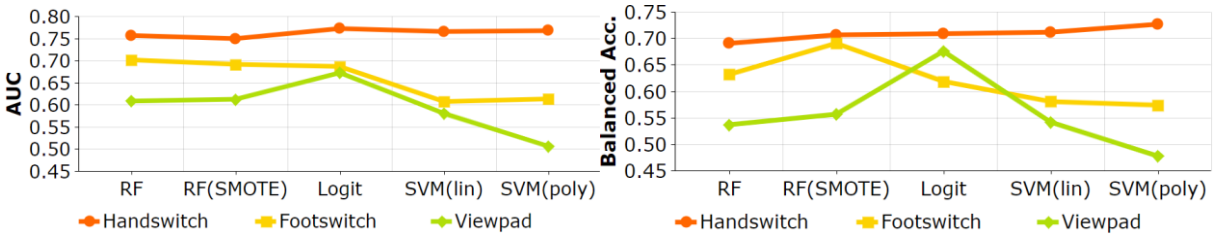


Figure 0.2 – AUC and balanced accuracy for binary classifiers

The graph above shows that no universally superior binary classifier was identified. Different binary classifiers were found to perform better for different stages in the cascade. The best binary classifier in terms of 'balanced accuracy' was selected for each stage. The 'threshold for labelling' of these binary classifiers was set such that an instance is only labelled to be of the 'positive' class if the model is very confident of this. The best performance was observed when a combination of both was used (0.444). The confusion matrix below describes the result.

Table 0.3 – UI modules results: confusion matrix - binary classifier cascade + XGBoost

		predicted				
		FootSwitch	HandSwitch	Other	ViewPad	Class.error
True	FootSwitch	39	28	2	4	0,466
	HandSwitch	14	98	8	5	0,216
	Other	13	21	9	3	0,804
	ViewPad	13	13	3	13	0,690

Translating these findings to practice, one would for a given case; 1) first apply binary classifiers until the case is predicted to be of the 'positive' class. 2) If the case is still labelled as 'Other' after applying all binary classifiers, one applies the multiclass classifier.

Depending on the wishes of the user and the alternatives at hand, one could also refrain from appending the multiclass classifier. Calls labelled as 'Other' could be investigated using other more precise methods using more specific data. The RCF prediction model might serve as a general first step, similar to the common analyzer tool.

It is difficult to say to what extend survival analysis can be used to improve the root cause analysis process in the maintenance of medical imaging systems, due to the lack of a widely accepted performance metric for the problem presented. For the 'BV Pulsera' and 'Brilliance CT 64 CHANNEL' systems, the root cause failure prediction modal based on survival analysis outperforms the benchmark for the investigated subsystems.

Preface

This thesis is the culmination of five years of studying and eight months of hard, but satisfying and enjoyable work at the Technical University of Eindhoven and Philips Research. I am grateful for the opportunity to finally bring my acquired knowledge into practice and contribute a little bit to the hardworking people at Philips, striving to improve the world through innovation.

Most of all I would like to thank my awesome supervisor, dr. Yingqian Zhang for her belief in me throughout the project and her endless support. You absolutely went above and beyond in helping me structure my progress and guiding me in the right direction, taking a considerable amount of time out of your busy schedule and sit with me to brainstorm about each and every single step in the process. I could not have done it without your support and I am incredibly grateful to you for your advice and expertise.

I would also like to extend my thanks to dr. Alp Akçay for the suggestions from your vast knowledge on the subject and providing additional guidance.

I am very grateful toward dr. Jan Korst, dr. Verus Pronk, dr. Mauro Barbieri, and the rest of my colleagues at Philips for your willingness to answer any question that I might have. I would like to thank Jan and Verus especially for guiding me throughout the process of finding my way, and for providing me with invaluable advice.

Finally, I want to thank my family, roommates, friends and fellow students. Special thanks to Jorick and Gregory, for struggling together with me with our academic work, always making sure that I am not slacking off. I would not have been able to do it without you guys.

Eindhoven, September 2019 Han Qi Xia

Contents

List of Figures

List of Tables

Part I – Introduction & Problem Description

1. Introduction

This report is the result of a master thesis project conducted at Philips Research, in collaboration with the Eindhoven University of Technology. The focus was on the possibilities provided by the data gathered on MISs (medical imaging systems). The ultimate goal of this project is to assess the applicability of survival analysis as a general tool to support the maintenance process of all parts and system types of MISs. Within the three relevant domains of 'survival analysis', 'system maintenance' and RCA (root-cause analysis), the following research gaps were identified:

- Use of survival analysis for RCA
- Multi-component system maintenance models dealing with censoring
- Data-driven RCA

As a primarily academic work, the focus of this project is the evaluation of a postulated RCF (root-cause failure) prediction model, based on survival analysis, in order to target the identified research gaps. As such, the main intended audience are the researchers at Philips and by extension, the service engineers in Best and around the world. This project is mainly orienting in its setup and provides an overview of where survival analysis based models provide value and the feasibility of implementing them, given the available data. The project might serve as the groundwork for researchers implementing a decision support system for service engineers at Philips, should Philips decide that the results of the project warrant further pursuit.

This report is divided into five parts, starting with the introduction and problem description. Within this part, the context of the project is explained. Information is provided on the company, data and stakeholders. The part is concluded with the problem statement of the project. In this section the goals, scope and methodology of the project are outlined and the research questions are presented. In the second part a literature overview is provided in which the main concepts for this research project are explained. The third part provides an analysis of the available data. Model features are identified and the extraction of them are described as well. The fourth part describes the modelling decisions taken during the implementation of the RCF prediction model. A conceptual model for RCF prediction based on survival analysis is suggested, specifically the KME (Kaplan-Meier estimate) and CoxPH (Cox proportional hazards) regression. The results are summarized and discussed in the fifth and final part. Recommendations and suggestions for further research are provided for Philips and future scholars, after which the report is concluded with the references and appendices.

2. Context: Philips Research

Starting in 1891 as a manufacturer of lightbulbs, Philips has since played an important role in the realisation of numerous inventions, such as the first electric shaver with rotating blades, the cassette tape and Compact Disk. The company has since consolidated its activities and committed itself to innovating health and well-being. Nowadays, Philips is one of the most innovative companies in the field health technology. Philips strives to make the world healthier and more sustainable through innovation, with the goal to improve the lives of 3 billion people a year by 2025. One of the ways through which Philips improves healthcare is through innovative MISs.

2.1. Data Science Department

Philips aims to maximize the predictability of the cost of ownership for anyone using its MISs. For this purpose, new systems have been designed with networking functionalities. These systems generate logs while in operation and this data is gathered and used by the data science department at Philips.

The department is responsible for delivering and maintaining predictive models and tools based on analysis and machine learning. These assets enable employees from all over the company to execute their tasks smarter and better. These assets make use of a diverse array of data and the latest innovations in data science. This is how the department contributes to the strategic goals of Philips, while focussing on sustainability and data management.

2.2. Maintenance contracts

When delivering a MIS, Philips offers to take care of the maintenance of the MIS to different degrees in the form of optional maintenance contracts. The customer may opt to have Philips take care of the maintenance process (partially), contract a third party, or perform the maintenance in-house. These contracts differ widely in their contents, some including the procurement costs of replacement parts in the overall costs, and some only covering the labour costs. Currently, the contract specifications are not subject to projected cost of ownership based on usage intensity analysis or survival analysis.

This fact also complicates the interpreting of the data regarding individual systems at Philips, as install dates become ambiguous. The most common scenario is that systems are brand new at the recorded install date. However, in some cases the systems has been operational before and maintained by a third party. This context becomes relevant in the accurate approximation of the age of parts, which is imperative for survival analysis.

2.3. Data availability

This research project follows a data-driven approach. The first vital step is the exploration of the available data. The ISDA (imaging system data analytics) data gathered by Philips is used to support a variety of data analytics projects running at Philips and can be classified into the following main categories:

Operational data encompasses logs regarding usage and performance measurements that have undergone an ETL (extract, transform & load) process. These data tables facilitate the analysis of usage and condition based maintenance policies.

Maintenance data of systems are stored in different tables in the collective database used by the data science department. Over the years, different storage solutions have been used. In each successive generation, slight differences are introduced in content and storage structure. Most experiments conducted during this project make use of data from the catmasterlist table which retired in late 2017. Four main components have remained largely unchanged for each different storage solution.

- Devices, Calls, Jobs and Parts

2.4. Service engineer corrective maintenance process

Maintenance is primarily classified as either PM (planned maintenance), FCO (field change order) or CM (corrective maintenance) in the catmasterlist database. This project focusses mainly on CM in this database, which accounts for a substantial part of the cases recorded.

CM can be initiated by a customer call or warning issued by predictive models. Remote monitoring engineers might also detect imminent failure and urge the customer to schedule a maintenance as soon as possible. It is however up to the client to heed the warning or not. Finally, field service engineers can initiate a CM when detecting imminent failure on site.

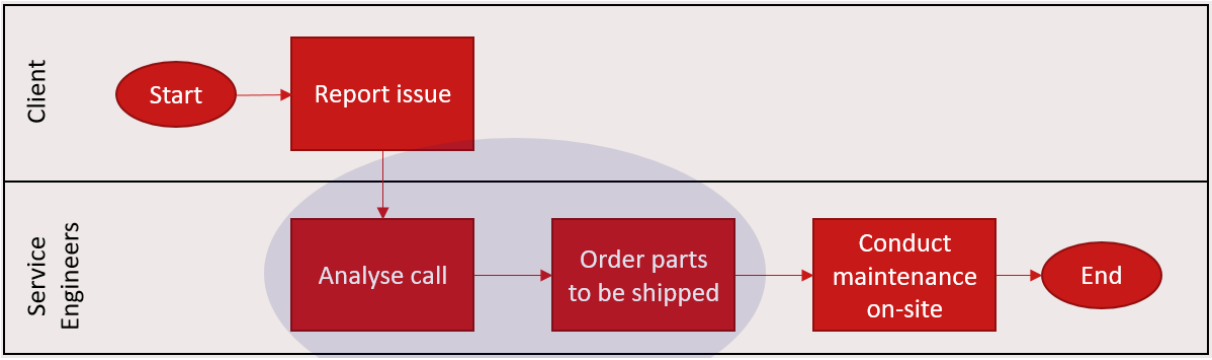


Figure 2.1 – Corrective maintenance call requiring part replacement: project point of focus

During this project, focus is place solely on CM. More specifically, only one case is investigated, namely CM calls which are solved by the replacement of a single part. Limiting the scope even further, this project researches methods to support the process of remote RCA and determining which parts should be shipped in order to maximize the probability of the right part being send, when the number of parts that can be send is limited, as shown in Figure 2.1.

3. Research problem

This section defines the problem that forms the impetus for this research project. Philips Research is investigating the application of survival analysis for the maintenance of MISs. First, the problem is defined. Subsequently, sub-problems, main stakeholders and research questions are described.

3.1. Problem introduction

Philips currently utilizes a wide array of different predictive models and is exploring condition-based maintenance policies for MISs. However, despite the advancements in the field of maintenance, corrective/unplanned maintenace still accounts for a majority of the maintenance performed. When taking into account the cost-benefit balance, CM will likely continue to make up a substantial portion of maintenance executed for the foreseeable future. There is also the issue of clients not heeding warnings from the remote monitoring engineer.

Philips has indicated interest in using survival analyses to improve the maintenance process of MISs. Data is continuously gathered while machines are operational. The problem is in the domain of maintenance engineering. Maintenance engineers are often interested in three closely related system attributes: reliability, availability and maintainability. When a system breaks down, high costs are generally incurred due to downtime of the system. This is especially the case in the domain of healthcare, where long waiting times are characteristic. Value can be gained by maximizing the availability, which is dependent on the reliability and maintainability of a system. Survival analysis has seen frequent use with the goal of measuring reliability. However, very little research has been done toward the applicability of survival analysis to improve the maintainability of system. The goal is to investigate the possibilities for improving the maintainability of multi-component systems with heavily censored data (numerous very reliable parts + short observation windows) using survival analysis.

3.2. Problem formulation

The problem that is analysed encompasses the difficulties that service engineers face when diagnosing the root cause of a failure when on-site CM is required. Service engineers are already supported by several data-driven tools, which have proven themselves very powerful for specific parts. However, systems are composed of hundreds or even thousands of parts, each with their own properties and failure distribution. Data availability is also limited for the more reliable parts and newer systems.

Currently, system engineers are supported by a ‘Common Analyser Tool’ when diagnosing the root cause of a failure where the cause is unclear. The tool loads the log data of the preceding days, on which log patterns are matched. The applicability of survival analysis in this process is a point of interest.

The medical domain is unforgiving for downtime and the MISs maintained by Philips are state of the art, and thus expensive. Each additional hour of downtime brings substantial costs with it. There is also the costs and difficulties associated with on-site part replacement. These are large complex machines with many expensive parts. This means it is unfeasible to bring a replacement for every part that might be the cause of failure. Shipping an expensive replacement part to the client, in case it is needed, leads to high inventory costs, as more spare parts need to be kept on hand.

The main goal of this project is to evaluate the applicability of survival analysis as a decision support system that aids the root cause analysis in complex systems that is able to deal with censored data. In order to be able to improve the maintenance process using survival analysis, it is first necessary to understand the way Philips maintains the MISs under its care. It is also necessary to evaluate the available data that is used to support the survival analysis. Thus, the following research question is asked:

How is the maintenance process of medical imaging systems currently supported by data gathered by Philips and can this process (involving any part) be supported further using survival analysis?

The defined problem can be split into two sections. First, the research focusses on analysing the current process and the data gathered by Philips. Secondly, a prototype implementation is tested and evaluated. After which recommendations for an ideal decision support system is formulated, providing direction for future research.

3.3. Research questions

To answer the formulated problem, several research questions have to be answered. The answers to these questions provide necessary insights. These research questions are described in more detail in this section:

Research question 1: What data relevant to Medical Imaging Systems maintenance is currently gathered by Philips?

A data-driven approach was used, which is explained further in Section 3.5. Thus, the first step is learning what data is available, as well as the quantity and quality of the data. Time and status of events are obvious necessities for the application of survival analysis. We are also interested in the presence of usable covariates such as the market, part type, system type configurations and usage of the system. This question was answered in part by conducting a series of interviews at Philips and exploring the data available regarding the maintenance of MISs in accordance with the CRISP-DM (cross-industry standard process for data mining) methodology. The results are presented in Sections 5, 6 and 7.

Research question 2: Can survival analysis be used to distinguish between heavily and lightly used individual systems of any system, using only part replacement data?

Both literature and domain knowledge indicate that usage is a powerful predictor for determining the survival rate of parts and systems. However, usage data is available for a limited number of systems. This question is asked in order to obtain an approximation of the 'usage intensity' covariate for all systems. The question was answered by applying non-parametric REDA (recurrent event data analysis), standard statistical tests and machine learning techniques, presented in Section 7. The latter approach is evaluated using data from systems whose usage is well documented, to determine whether survival analysis can be used to reliably predict whether a system is 'heavily used' using only the replacement data.

Research question 3: Can survival analysis methods be used to support root-cause analysis at Philips?

In order to answer this research question, it is necessary to evaluate the performance of a prototype implementation. Scientific validity and proper conduct dictates that a benchmark against other techniques is to be performed.

In pursuit of this goal, a conceptual model is postulated. This conceptual model describes the inputs, inner workings and outputs on a highly aggregated level. It also describes the interpretation and evaluation of the outputs. The concept is based on literature, expertise from data scientists and service engineers from Philips, and the expertise from machine learning experts at the TU/e. This conceptual model is realized in an experimental prototype, which is used to answer the research question.

Research question 4: To what extent are the results of the analysis answering the third research question dependent on the system and part type in question, in regards to data availability?

The goal is to eventually obtain a general method of improving the maintenance of multiple parts and modalities. Maintenance data is recorded very similarly across different modalities, but data availability differs for different system types. The performance of the algorithm is dependent on the data availability. Thus, it is necessary to measure how this affects the accuracy of the model. How long should the parts in a system have been in use, or how many historic cases should have been recorded, to reliably estimate its failure behaviour? Is there a threshold regarding data availability? Does the performance of the algorithm drop for parts that are rarely replaced? And by how much? The answer is primarily derived from the evaluation of the prototype implementation and a separate sensitivity analysis based on partially artificial data where additional censoring and removal is applied, presented in Section 10.

3.4. Scope

Several decisions have been made to adjust the magnitude of this research project. An overview of these decisions is given in this section.

3.4.1. Modalities, system types and parts

The goals is to support the maintenance of all parts and modalities that Philips maintains. But testing the performance of the model for each and every part is infeasible when taking into account time constraints. The models are tested on a limited, but diverse set of (sub) systems.

3.4.2. Non-parametric and semi-parametric survival analysis methods

Survival analysis or time-to-event data analysis are broad terms encompassing a wide array of techniques, each with their own nuances, applicability and interpretation (Kalbfleisch & Prentice, 2011). This research is limited to researching the performance of non-parametric and semi parametric methods. Including parametric methods would lengthen the project length to beyond 6 months. This decision is based on the high amount of censoring observed among the data. The ability of non-parametric methods to deal reliably with censored data has been proven on numerous occasions (Arthur & Peterson, 1977).

3.4.3. Implementation

In order to maximize the applicability of this research, preference is given towards using real company data over simulated data where possible. The use of 'usage data' is limited to validation of the *usageIntensity* approximation method. It is not used as input for the model in order to keep the method applicable to any kind of system, for which part replacement data is available. Systems are treated as multicomponent systems, with some exceptions where REDA is applied.

A prototype implementation was created in order to be able to evaluate the conceptual model. The prototype implementation is limited to a proof of concept. On site usage test by service engineer is left out of scope. This project only provides future directions for a possible integrated decision support system that supports the service engineers of Philips.

3.5. Research design

As stated before, this research is a case study at Philips Research. The research design of this project and the methodology followed are explained in this section. The necessary steps needed to answer the research questions are described together with the methods used to verify the value of the results of the project. This project was performed in adherence to the CRISP-DM methodology. A diagram of the main steps in this process is depicted in Figure 3.1.

Figure 3.1 – CRISP-DM diagram (Shearer, 2000)

Part II – Literature overview

4. Literature overview

In line with the objectives of this research, this overview first describes multiple popular and proven non-parametric survival analysis methods. Subsequently, relevant literature on root-cause failure analysis is described. Finally, the section is concluded with an overview of multi-component system maintenance models, with their characteristics and properties.

4.1. Non-parametric Survival analysis methods

Parametric models haven proven themselves powerful tools, outperforming non-parametric methods in efficiency when their distributional assumptions are met (Gamel & Vogel, 1997; Peng & Carriere, 2002). However, these circumstances are the exception, rather than the rule. The research project will require the fitting of models to hundreds of parts with different distributions. It would be a great boon if the method does not require the determination of an appropriate distribution.

More importantly, there is the dataset at hand to consider and its properties. The maintenance data on the MISs are heavily censored. Aljawadi, Bakar, & Ibrahim (2012) compared non-parametric and parametric survival models using simulated data and found that the bias of parametric models exceeded that of non-parametric model as censoring rates increase. As can be seen in below, less bias was observed for parametric model when censoring rate was low. Non-parametric were found to be more consistent as censoring rate increased beyond 30%. A censoring rate well over 30% was observed during the data exploration phase in this project, as few parts were observed to fail more than twice.

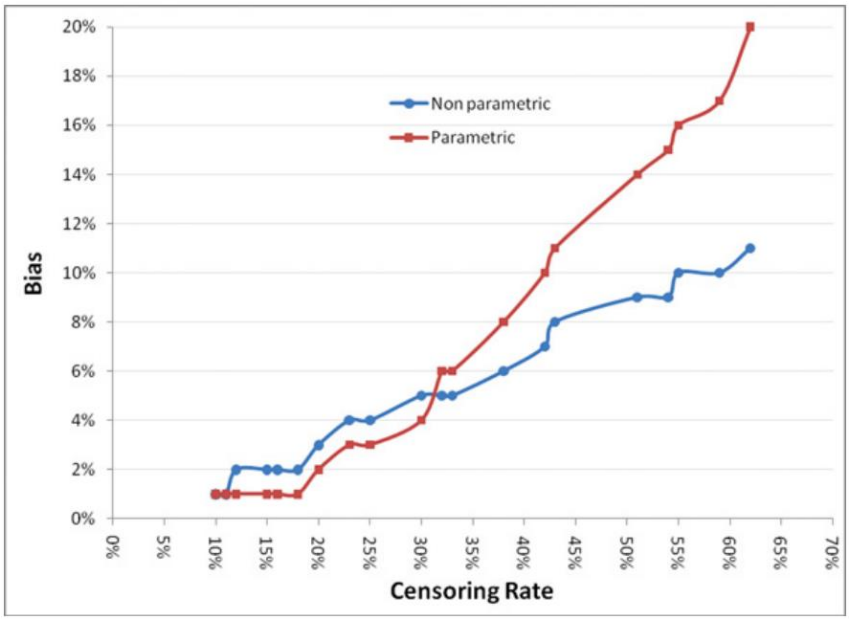


Figure 4.1 – Censoring rates vs bias based on simulation results (Aljawadi et al., 2012)

4.1.1. Kaplan-Meier product limit estimator

Kaplan & Meier (1958) proposed a standard estimator of the survival function, which is obtained by taking the product of a sequence of conditional probabilities. Uncensored events refer to those individuals for which the event of interest is observed within the window of observation. In this project, the event of interest is the failure of a part. Events are censored, if the failure was not observed before the end of the window of observation, or a part was preventatively replaced. The formal definition of the KME is given below:

$$\hat{S}(t) = \prod_{i:t_i \leq t} \left(1 - \frac{f_i}{r_i}\right), \tag{1}$$

where the t_i's are the ordered observed survival times or censoring times from the sample, f_i is the number of uncensored events that occur at a time t_i, and r_i is the number of individuals at risk at t_i.

For a given population without censored events, the KME is the proportion of individuals that have survived until time t. When censored events are present, the calculation of the KME is less straightforward. Censored events influence the shape of a Kaplan-Meier curve in the following way:

1. When one or more censored events occur and no completed events, the curve is not adapted. $f_i = 0$ in this case, which means that regardless of the change in r_i, $\frac{f_i}{r_i} = 0$. Thus the current survival rate is multiplied by 1, meaning it does not change.
2. The next time a completed event is recorded, the previous censored event are exerting influence on the change of the curve.

4.1.2. Cox proportional hazards regression

In 1972, Cox proposed a multi-variate semi-parametric model that specifies the hazard rate $h_i(t)$ of an individual i with covariate vector z_i. The formal definition of $h_i(t)$ is given below:

$$h_i(t) = \exp(\alpha(t) + \beta * z_i), \quad t \geq 0, \tag{2}$$

where β represents a vector of regression coefficients, $\alpha(t)$ represents a baseline hazard rate, which is an unknown nonnegative function of any form, making the model semi-parametric. The statistical problem entails the estimation of β and $\alpha(t)$, using a dataset of n observed individuals with covariate vectors $z_1, \dots, z_n$ and times of event $T_1, \dots, T_n$. Over the years, many methods of inferring this β were suggested by different researchers.

Fitting the models/estimating the regression coefficients

Cox himself suggested the following partial likelihood function for the inference of the optimal regression coefficients vector $\boldsymbol{\beta}^*$ in 1972:

$$L(\boldsymbol{\beta}^*) = \prod_{i=1}^{n} \left\{ \frac{\exp\left(\boldsymbol{\beta}^* z_i(T_i)\right)}{\sum_{j \epsilon R_i} \exp\left(\boldsymbol{\beta}^* z_j(T_j)\right)} \right\}^{d_i}, \tag{3}$$

where d_i is an indicator for censoring and $R_i = \{j : T_j \geq T_i\}$ are sets of all individuals still at risk at T_j. The inferred optimal regression coefficients vector $\boldsymbol{\beta}^*$ is the $\boldsymbol{\beta}$ that maximizes (3). This method was not used during this project due to long computation times for large datasets.

Instead, the method proposed by Andersen & Gill (1982) was used, who suggest inferring $\boldsymbol{\beta}^*$ using a counting process formulation with local martingales. Given a time interval [0, 1], a sequence of models are considered indexed by $n = 1,2, \ldots$. Instead of observing possibly censored lifetimes of n individuals, as explained above, we generalize as follows:

We observe n m-component counting processes $N^{(m)} = N_1^{(m)}, \ldots, N_n^{(m)}$ where each counting process $N_i^{(m)}$ counts the observed evens of the ith individual, $i = 1, \ldots, n$, over the time interval [0, 1]. The sample paths of each counting process are step functions with jumps of size +1 only.

It is assumed that each $N^{(m)}$ has a random intensity process $h^{(m)} = h_1^{(m)}, \ldots, h_n^{(m)}$ such that:

$$h_i^{(m)}(\mathrm{t}) = Y_i^{(m)} * \exp\left(\alpha(t) + \boldsymbol{\beta} * Z_i^{(\boldsymbol{m})}(t)\right), \tag{4}$$

Where $\boldsymbol{\beta}$ represents a vector of regression coefficients and $Y_i^{(m)}$ is a predictable process taking values in {0,1} indicating when the ith individual is under observation. So $N_i^{(m)}$ only jumps when $Y_i^{(m)} = 1$. $Z_i^{(\boldsymbol{m})} = \left(Z_{i1}^{(\boldsymbol{m})}, \ldots, Z_{ip}^{(\boldsymbol{m})}\right)'$ is a column vector of p covariate processes for the ith individual. Let $C(\beta, t)$ be the logarithm of the Cox partial likelihood (3), evaluated at time t. Given (3) and (4) we state that:

$$\begin{aligned} C(\beta, t) = \sum_{i=1}^{n} \int_0^t \boldsymbol{\beta} * Z_i^{(\boldsymbol{m})}(s)\, dN_i^{(m)}(s) \\ - \int_0^t \log\left\{ \sum_{i=1}^{n} Y_i^{(m)}(\mathrm{s}) * \exp\left(\boldsymbol{\beta} * Z_i^{(\boldsymbol{m})}(s)\right) \right\} d\bar{N}(s), \end{aligned} \tag{5}$$

where $\bar{N} = \sum_i^n N_i$. Then $C(\beta, 1) = logL(\beta)$, and the $\boldsymbol{\beta}^*$ is defined as the optimal regression coefficients vector that maximizes $C(\beta, 1)$

4.1.3. Nelson-Aalen non-parametric estimator for Recurrent Event Data Analysis

Another non-parametric estimator was proposed by Nelson (1972) and evaluated by Aalen (1978). This estimator of the hazard has been frequently used and is commonly referred to as the Nelson-Aalen Estimator and is given by the following formula.

$$\hat{H}(t) = \sum_{i:t_i \le t} \left(\frac{f_i}{r_i}\right), \tag{6}$$

where the t_i's are the ordered observed survival times or censoring times from the sample, f_i is the number of events that occur at a time t_i, and r_i is the number of individuals at risk at t_i.

This estimator can be used to compute the MCF (mean cumulative function) as described by Nelson (2003), allowing the exploration and modelling of recurrent event data. One takes the mean of multiple cumulative event functions of a population of individuals, such as repairable systems at a point in time. Plotting the means over all points in time returns the Nelson-Aalen estimator of the cumulative hazard function.

4.2. Data driven root-cause failure analysis

RCA in maintenance decision-making is the process of identifying the factors that cause system failures. The paradigm currently observed in RCA for maintenance decision making is characterized by primarily qualitative or semi-quantitative methods (Chemweno et al., 2016). These methods introduce the issue of unavoidable bias of experts. All but a few quantitative methods become computationally infeasible as the number of components in the system grows beyond a fraction of what is usual for most MIS subsystems.

Despite the growing availability of machine maintenance data, there is barely any literature that leverages this data for decision support in RCA in maintenance (Chemweno, Pintelon, Van Horenbeek & Muchiri, 2015). From the scarce literature on data-driven RCA, the work of Zawawy et al. (2012) bears some relevance to the given problem. A data-driven RCA method using markov-logic networks, which provides ranked diagnoses, is described. Figure 4 below gives an overview of the diagnostic process. The overall structure of this method is one where SQL is used to extract data that is used and the output is a list of possible RCF ranked on their probability calculated by the underlying logic. The framework of this method provides some guidance for designing the RCF prediction model.

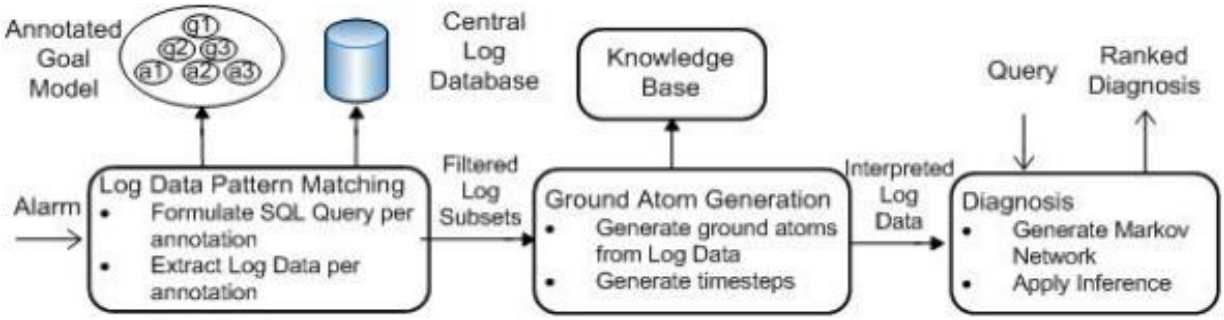


Figure 4.2 – Requirements-Driven RCA: Diagnostic Process (Zawawy et al., 2012)

4.3. Multi-component system maintenance models

Nicolai & Dekker (2008) define three different dependencies that require attention in multi-component system maintenance models. These are stochastic, structural and economic dependence. Stochastic dependence is when the failure of one part affects the failure distribution of another part. Structural dependence refers to when components structurally form a part or subassembly in a way that maintenance of a failed component implies maintenance on working components. Economic dependence refers to instances in which grouping replacements of multiple parts saves costs. Multiple researchers have tackled multi-component systems maintenance and proposed policies. Of these researchers, only a handful have used real data or handled censoring. Arbitrary stochastic, structural and economic dependencies are often assumed.

Van Horenbeek & Pintelon (2013) describe a dynamic policy for multi-component system maintenance. The basic premise of the policies is centred on the existence of economic dependencies and the possibility to save cost by grouping maintenance of multiple components. Grouping decisions are made dynamically, applying a dynamic programming algorithm to determine the optimal grouping structure for each point in time within the time horizon.

The policy consists of the following steps:

- Prediction of remaining useful life by estimation of the failure probability function
- Individual maintenance optimization by decomposition and construction of a tentative maintenance plan
- Calculation of penalty functions
- Maintenance activities grouping
- Maintenance execution and rolling-horizon update

Van Horenbeek & Pintelon (2013) tersely describe a dependence parameter α_d, which should reflect the economic and structural dependence of multiple components. Unfortunately, obtaining such a parameter is no trivial task and requires significant assumptions. Given the complexity of MISs, the presence of this step heavily reduces the feasibility of applying the proposed maintenance policy.

This overview has provided insight into some non-parametric survival analysis that are proven accurate in an environment with a high degree of censoring. The available data is explored in the following section.

Part III – Data exploration, understanding & preparation

5. Data exploration

A deeper understanding of the data is obtained during this phase. The available database is explored in order to understand what data is available, as well as its quality. This exploration is guided by the problem formulation. The focus lays on data that is needed for survival analysis, as well as data that could function as covariates.

5.1. Variables of interest

The maintenance databases contain records of 'calls'. Each call has a *CallOpenDate* and *CallCloseDate*, which are timestamps indicating the start and end of calls. Each case also contains a *CallType,* which denotes whether the case is one of CM, PM or other. This data is of particular interest as it can be used to obtain survival analysis specific variables in the data preparation phase as explained in Section 1.2. The variables that are extracted from the maintenance database are described below in Table 5.1.

Table 5.1 – Description: data of interest from the maintenance databases

5.2. Current and legacy databases

The decision is made to explore the newer [redacted] database in conjunction with the [redacted] database, despite the added complications. Cases from different databases can be linked using persistent identifiers. Unresolved issues with some of the data in the [redacted] database are known. These are tackled in the data preparation phase.

5.3. CallType frequencies

The frequencies of the different *CallType* for replacement calls are shown below in Table 5.2.

[redacted]

Table 5.2 – CallType frequency and percentage of replacements calls

[redacted]

6. Data preparation

In this phase of the CRISP-DM methodology, data is extracted, cleaned and transformed in order to obtain a final dataset that can be used for analysis and modelling.

6.1. Data extraction

Analytics data supporting the maintenance and operations of MISs are gathered in a Vertica database solution for machine learning and advanced analytics. Data extraction is facilitated by the JDBC (java database connectivity) API (Urbanek, 2018b), which allows the accessing of the database using SQL queries. The majority of the modelling in this project is implemented in the R programming language. The 'rJava' package (Urbanek, 2018a) is used to enable the calling of java methods in R.

6.2. Data aggregation

Data from different generational databases are aggregated using persistent (mostly unique) identifiers. Several calls required the removal of duplicate data resulting from combining databases. Some calls are recorded with different timestamps as can be seen in Figure 6.1 below. [redacted]

[redacted]

Figure 6.1 – Illustration of cases requiring data aggregation

6.3. Feature selection

Arguably even more important than model selection, feature selection plays a vital role in statistical learning. Keeping the idiom 'garbage in, garbage out' in mind, an algorithm is guaranteed to give nonsense output when given nonsense input data. Hence, relevant features are identified and constructed in the following sections.

6.3.1. Readily available features – applying the Log-rank test

Several categorical features could be directly extracted without any feature engineering. These were identified by applying the KME with one grouping variable and applying the Log-rank test. These features are described in Table 6.1 below:

Table 6.1 – Readily available features

Feature	Description
[redacted]	

Figure 6.2 below depicts the KME survival curves of [redacted] belonging to the [redacted] *ReplacementChain* within system types whose configuration [redacted]. These survival curves are grouped per *MarketGroup*. The NAM (north America) strata shows the highest survival rate by far, with the median occurring after around [redacted] . The GRC (greater China) strata shows the lowest survival rate, with a median survival rate after around [redacted]. The Log-rank test returns a p-values smaller than 0.0001, indicating a statistically significant difference between the groups.

Figure 6.3 shows the survival curves of parts belonging to the [redacted] *ReplacementChain*. These survival curves are grouped per *Part12NC*. Statistically significant differences were observed between the three groups. The configuration with *Part12NC*=[redacted] appears to have the highest survival rate.

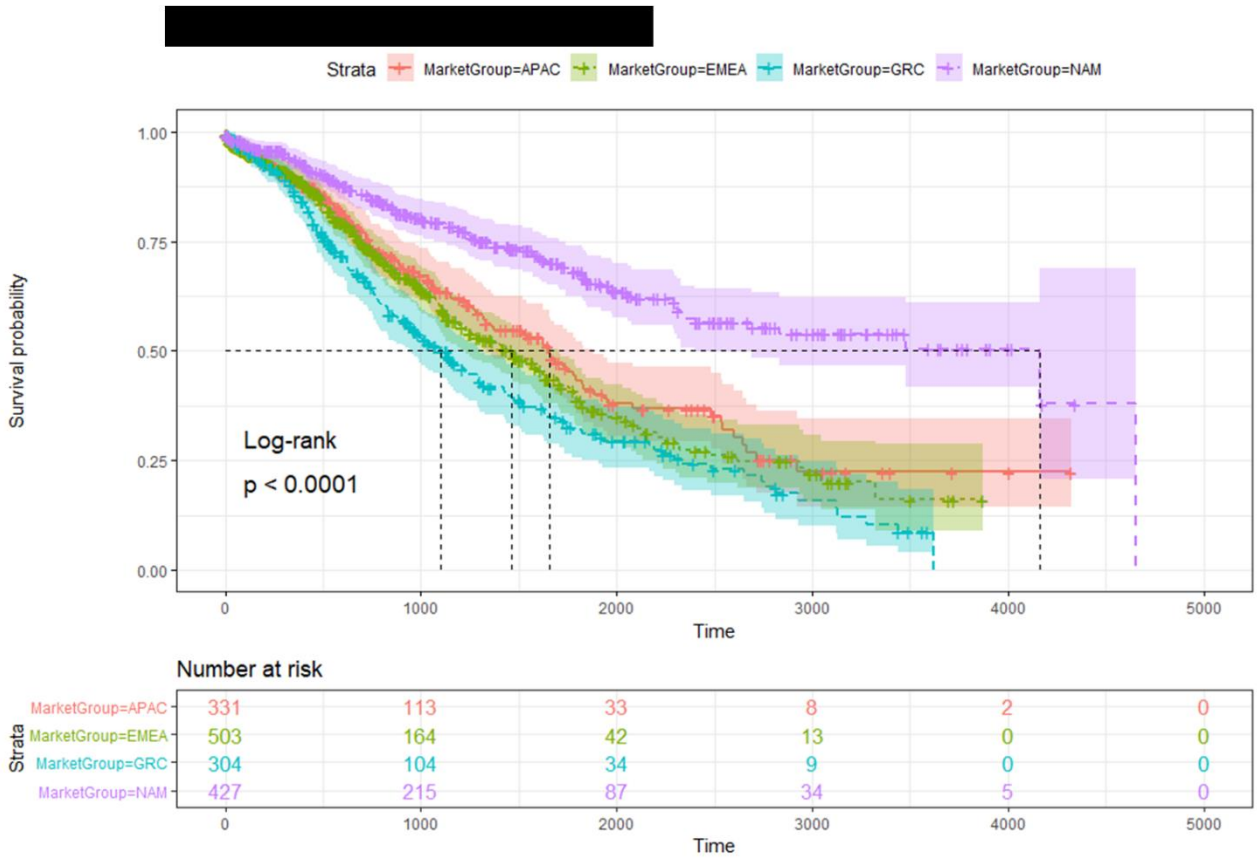


Figure 6.2 – Survival curves by MarketGroup: ReplacementChain=[redacted]

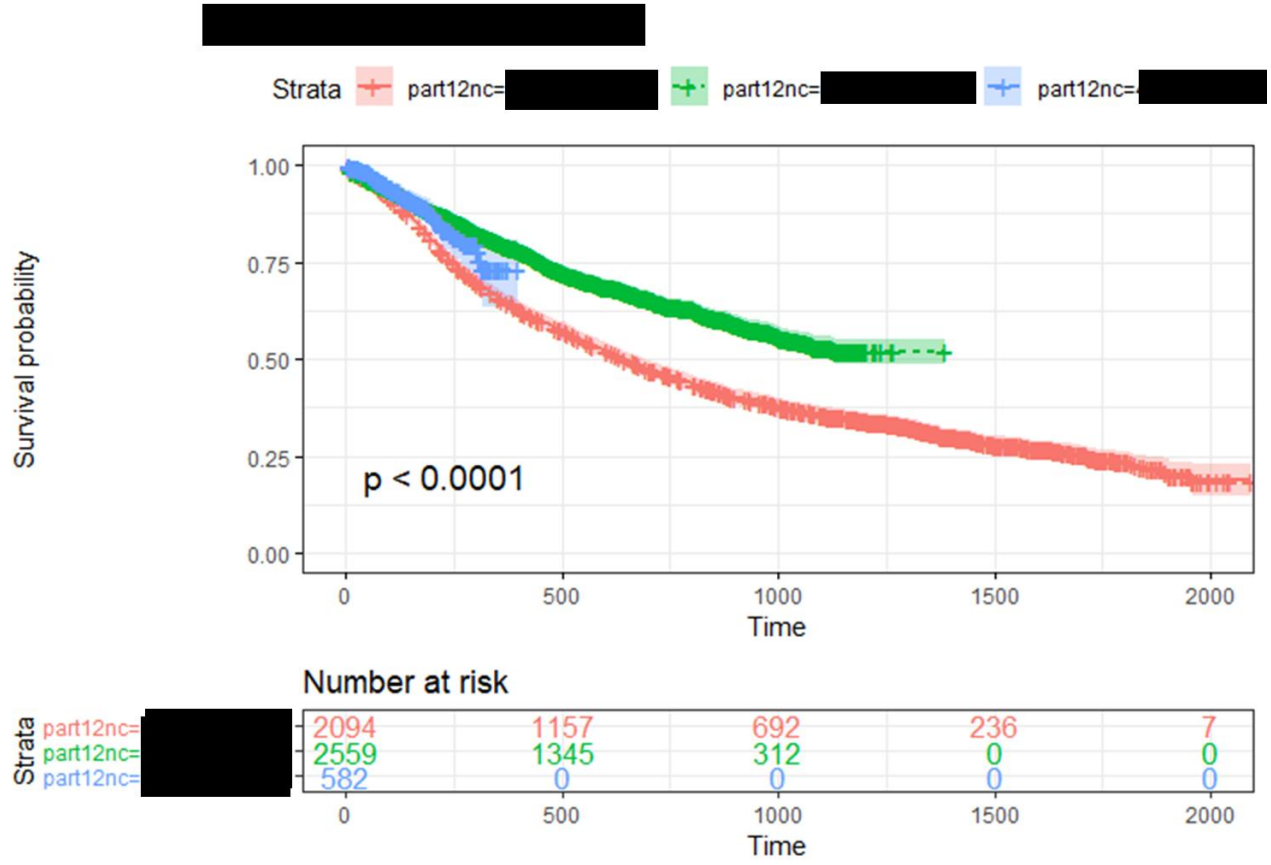


Figure 6.3 – Survival curves by Part12NC: ReplacementChain =[redacted]

As shown in Figure 6.4, no statistical significant difference was found between two similar and relatively new system types [redacted]. A statistically significant difference was found between these systems and a slightly older system type [redacted]

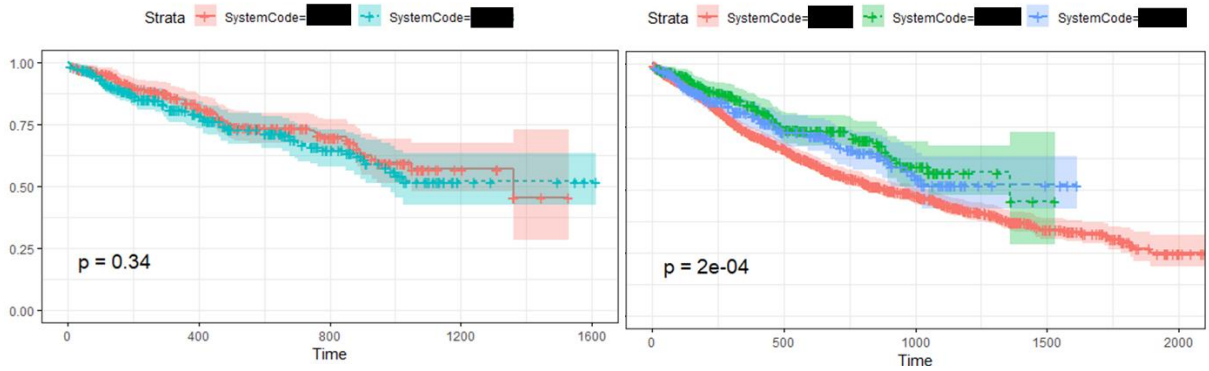


Figure 6.4 – Survival curves by SystemCode: ReplacementChain=[redacted]

1.1.1. 'Usage intensity' feature

Domain knowledge, previous projects conducted at Philips and literature support the notion that usage intensity of machines is linked to the failure distributions of some parts. Several systems are equipped with software that gathers usage data. However, these are generally only the newer and more advanced systems. Data suitable for data analysis is gathered for only a small portion of the systems maintained. In contrast, maintenance data is much more abundant. Thus, we investigate whether the usage of machines can be approximated using only replacement data. This process is explained in detail in Section 7.

1.2. Survival Analysis specific data preparation

In order to prepare the data for survival analysis, several assumptions are made.

- **Events (repair or replacement) are treated as a point in time**. This assumption is justified by the ratio between the time between events and the mean duration of a repair. The latter rarely exceeds a length of 24 hours. Due to this data characteristic, choosing between *CallOpenDate* and *CallCloseDate* is of little consequence.
- ***CallOpenDate* is used as the time of event**. There are multiple options to define the exact time of the event of interest and extract it from the data. *CallOpenDate* is preferred over *CallCloseDate* as some cases are left open indefinitely for administration purposes.
- **Repairs are assumed to be perfect.** A part is treated as new following a repair.

6.4.1. Survival time and status variables

The core variables for survival analysis are the 'survival time' and 'status' variables. 'Survival time' is defined as the time interval between A and B for part replacements. A is the moment a new part is placed into the system. B is the time where the event of interest (failure of a part)

occurs or the part is no longer observed (censored). In the database, *CallOpenDates* are recorded as calendar dates, e.g. yyyy-mm-dd. This is referred to as secular times in survival analysis literature. These calendar dates are aligned to obtain the survival times. This process is illustrated below in Figure 6.5 with the survival times of 6 parts. Parts are assumed to have been observed starting from the same point in time, while the duration is kept the same.

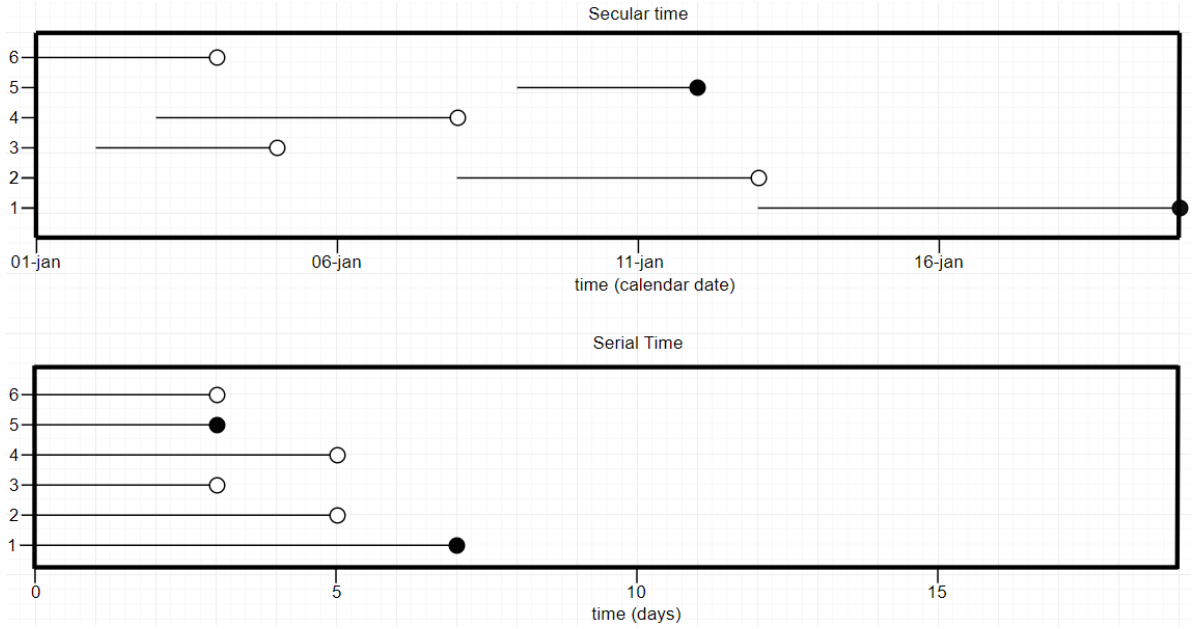


Figure 6.5 – Time variable: data preparation

The *CallType* of the subsequent data entry (maintenance call involving the same part) is used to determine the status of a given event. When defining part replacement as the event, an event is uncensored if the subsequent data entry has a *Quantity* of 1 and its *CallType* is CM. When defining the event as a repair of a part in the system, we do not check *Quantity*. This definition relies on the assumption that repairs are perfect.

6.4.2. Approximating time window of observation

When applying survival analysis, there is also need to know the window of observation. This window of observation is different for every system in the ISDA database. When the maintenance database no longer contains any entries for a specific system after a point in time, it is assumed that the system was taken out of operation or that the database was no longer being filled with new data.

General non-replacement maintenance calls regarding any part are recorded consistently for individual systems. Even when the install date of the system is known, it is not possible to use the install date of a system as the starting point of the observation window. As mentioned before, Philips does not maintain all MISs produced. Systems might have been operational long before Philips started maintaining it. There might have been unknown replacements of parts before this time window. In order to approximate the time window of observation, latest call date of a system is assumed to be the end date of the window of

observation. For part specific survival data, the start of the window of observation is set as the first maintenance call recorded in each individual system.

6.4.3. Observing the entire *ReplacementChain*

As mentioned before, parts are constantly innovated and new designs constantly replace older designs. These designs are labelled with their own unique *Part12NC*. When computing the survival curves of a specific part, it is necessary to include data detailing the entire *ReplacementChain* of that part. In the Teradata_OneEMS maintenance database, both the part that is being replaced and the replacing part are recorded. However, in the legacy databases, only the replacing part is recorded. This is problematic in the scenario when a failed part is 'forward' replaced, e.g. a part of an older design is replaced with a part with a new *Part12NC*. The failure and subsequent replacement of that part would go unnoticed, if one were to only query *Calls* for which the recorded *Part12NC* is equal to the *Part12NC* in question.

To illustrate this issue, observe the survival curves of the *Part12NCs* within the *ReplacementChain*=[redacted] plotted in Figure 6.3 above. Around 2015, a new design was introduced for the [redacted] with a new *Part12NC*=[redacted]. From this point in time onwards, parts belonging to this *ReplacementChain* would be replaced with this new design. The graph on the left in Figure 6.6 shows the survival curve obtained from a dataset obtained where only calls where *Part12NC='* [redacted] were queried. This survival curve would lead one to believe that these parts suddenly stopped failing after a certain age. This was due to the old design no longer being used. Replacement calls would be recorded in the database with *Part12NC*=[redacted]

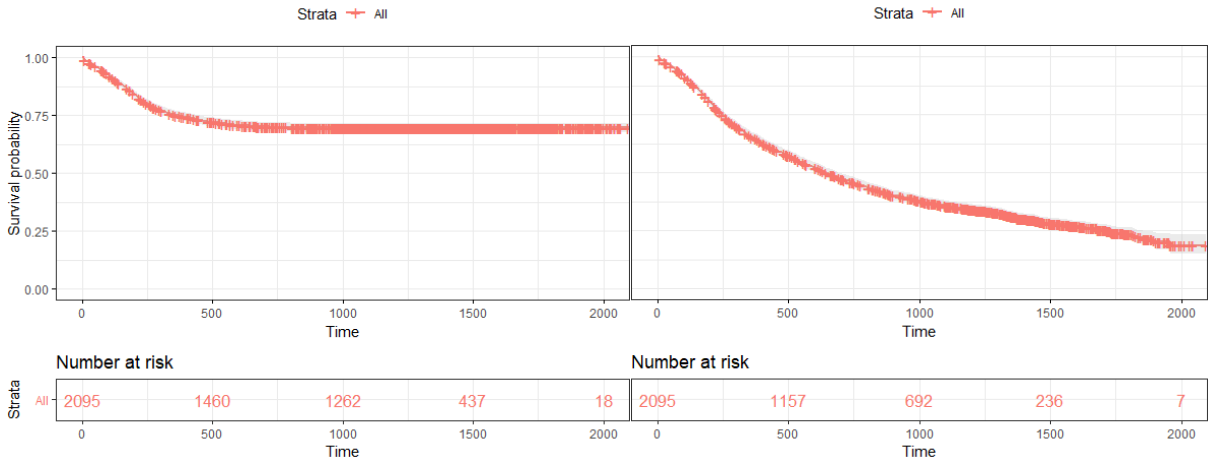


Figure 6.6 – Survival curves from (in)complete queries: Part12NC=[redacted]

Thus when estimating survival rate of part with specific *Part12NC*, calls relating to the entire *ReplacementChain* should be queried. The graph on the right shows the survival curve resulting from such a dataset.

7. *UsageIntensity* approximation

It is reasonable to assume that parts in a more heavily used system will fail more frequently. Because of this, the decision is made to include a *usageIntensity* variable that approximates the extent to which systems are used. Remember that the objective of this project is to obtain findings applicable to all system types and parts. Thus the following condition is imposed: the *usageIntensity* variable should be obtainable for all system types.

This section explains how the *usageIntensity* covariate is obtained. Standard machine learning techniques are used to train a model that predicts the *usageIntensity* of a system using only universally available data. A training dataset is obtained from a set of systems for which high quality usage data is available. For this analysis, only data from systems with the *SystemCode* [redacted] are used, which boast the most extensive records in the database. Limiting the data used to a single market was also considered, as it could possibly ensure higher uniformity in procedures. Nevertheless, this advantage is outweighed by the disadvantage of having significantly fewer data instances.

7.1. Exploring 'usage' data

Data on the usage of systems is derived from [redacted] table. Initially, the number of distinct *ExamIDs* was investigated as an option to calculate *usageIntensity*. *ExamIDs* consist of the system identifier and a timestamp corresponding to the start time of an exam, separated with an underscore e.g. '100713_2013-04-05_14:00:26'. Several other data were considered, such as the number of table movements or core component usage. When considering a system as a whole, using distinct *ExamIDs* was deemed the most suitable in terms of plausibility of the required assumptions and universality across modalities. Extracting this variable is also the least computationally intensive method.

Using *ExamIDs* implies the assumption that each individual exam places the same amount of stress on the system and causes the same amount of wear. This assumption is easily shown to be erroneous, as individual exams vary in duration and the number of parts involved. This is especially the case for the iXR modality where exams provide surgeons a real-time image for upward to several hours, instead of snapshots/single images. The *usageIntensity* approximation models used during the RCF prediction model evaluation in Sections 8 & 9, do not use the *ExamIDs* variables. Instead, $usageDuration$ is used as explained in Section 7.2. For now in this section, Section 7.1, *ExamIDs* are used to simplify the data exploration and keep the findings applicable to modalities other than iXR.

7.1.1. 'Gaps' in usage data

The number of unique *ExamIDs* recorded for the system with *EquipmentNumber* [redacted], plotted against time is shown in Figure 7.1. The vertical red lines indicate gaps in the data.

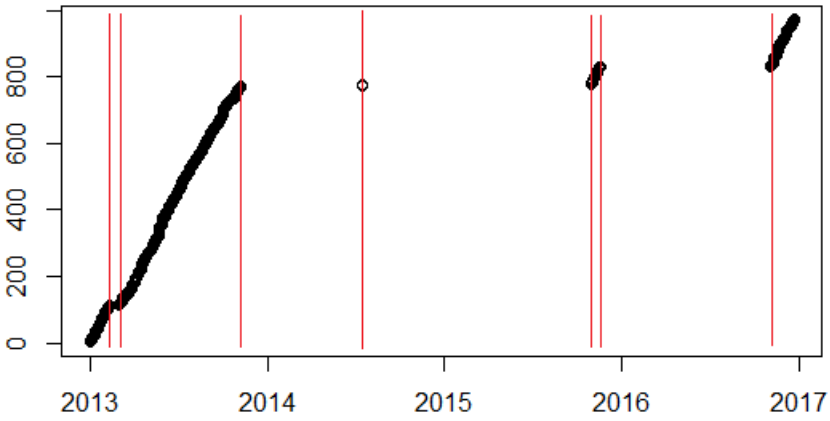


Figure 7.1 – Number of distinct ExamIDs for System [redacted]

Several properties can be observed from this figure. One observes that the time between consecutive examinations is small and consistent. For this specific system, roughly three or four examinations are conducted per day. The rate of examinations occurring per day is stable in the periods without gaps, of which multiple are observed in the data. Several explanations seem plausible such as the obstruction of networking capabilities of the system. Figure 7.2 shows that gaps in usage data do not overlap with gaps in maintenance data. It is assumed that machines are continuously used, even during the gaps. These 'gaps' are formally defined as periods in which no examinations have been recorded for over a week.

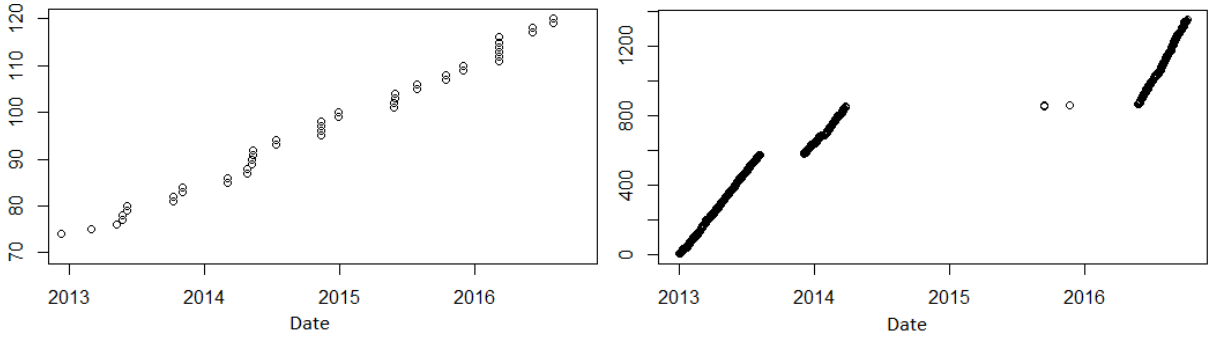


Figure 7.2 – Maintenance and usage data of the system with ConfigId= [redacted]

Usage data is aggregated per day. For each individual system, we are considering a window consisting of N consecutive days, during which at least one examination has been recorded. Let X_i be the number of distinct *ExamIDs* recorded for an individual system at the ith day, $i = 1, \ldots, N$. Let $t_1, \ldots, t_N$ be the corresponding dates. We wish to omit, in time, large gaps between successive examination start times. To this end, we define, for $i = 1, 2, \ldots, N-1$, Gap_i as follows:

$$Gap_i = \begin{cases} 0, & t_{i+1} - t_i \leq 7\ days \\ t_{i+1} - t_i, & t_{i+1} - t_i > 7\ days \end{cases} \quad (7)$$

For each individual system, $usage$ is now defined as:

$$usage = \frac{X_N}{t_N - t_1 - \sum_{i=1}^{N-1} Gap_i} \quad (8)$$

For each individual system, a $repair$ variable is calculated, which represents the number of **corrective replacements** ($Quantity = '1'$) occurring per year for the system during the window of observation:

$$repair = \frac{NumberOfCorrectiveReplacements}{Max(CallOpenDate) - Min(CallOpenDate)} \quad (9)$$

As an alternative, the number of **corrective repairs** ($Quantity \in \mathbb{N}_0$) was also investigated. The use of this predictor resulted in a decrease in model performance and thus replacements are used henceforth. This observation could possibly be attributed to the higher variance of 'repair' calls. Client behaviour is less consistent for repairs than replacements. Some clients call more often for small issues, while some clients will attempt to fix the issue themselves first.

In Section 8 & 9, a different variable, $repairRatio_j$, is used for analysis. We denote $repair_j$ as the $repair$, as described in (9), of a system j divided by the average $repair_j$ of all M systems with the same system type. This $repairRatio_j$ variable of a given system $j = 1, \ldots, M$, is calculated as follows:

$$repairRatio_j = \frac{repair_j}{\sum_{j=1}^{M} \frac{repair_j}{M}} \quad (10)$$

Where M is the number of systems with the same *SystemCode*. While not used in the Subsection 7.1.2, this variable is used as a covariate and feature in Sections 8-10. The use of $repair$ was deemed as more intuitive for the exploration of the variables.

Also important to note is the presence of several systems for which data is recorded for an extremely short period of time, such as a single day. These outliers were removed.

7.1.2. Describing 'usage' & 'repair' variables

After calculating the defined variables, their distributions are analysed. Testing whether the data is normally distributed is often used as a basic first step in analysing a variable. The Shapiro-Wilk test is used to test whether either variable is normally distributed. Very low p-values were returned for the *usage* (2.2e-16) and $repair$ (5.885e-13) variables for the [redacted] systems. Thus, we conclude that the data is not normally distributed, a notion that is supported by the flaring tails observed in the theoretical quantiles plot in Figure 7.3 below.

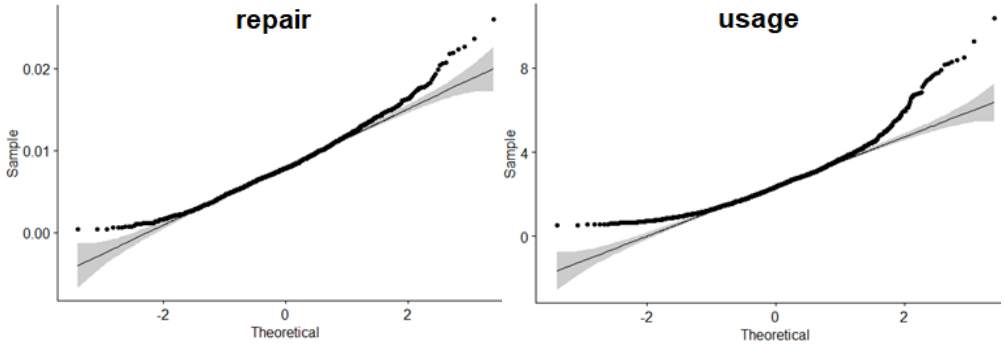


Figure 7.3 – QQ-plot usage & repair: SystemCode=[redacted]

In order to get some idea of the distributions of the variables, the histograms are plotted in Figure 7.4 and Figure 7.5. This visual data reconfirms the lack of a normal distribution. Basic descriptives of the data recorded for the [redacted] and [redacted] system types are given in Table 7.1 and Table 7.2. One notices that the distribution of the usage and repair variables for the newer [redacted] system type are more alike. Both display a similar skew and kurtosis score.

Table 7.1 – Descriptives usage & repair: SystemCode=[redacted]

Figure 7.4 – Histograms usage & repair: SystemCode=[redacted]

Table 7.2 – Descriptives usage & repair: SystemCode=[redacted]

Figure 7.5 – Histograms usage & repair: SystemCode=[redacted]

Correlation between usage and repair data

One of the simplest methods of determining the usability of 'repair' as a predictor for 'usage' is to measure the correlation between the two variables. Table 7.3 below shows the results of three widely used correlation tests, together with the p-values.

Table 7.3 – Correlation usage & repair: SystemCode[redacted]

Method	Correlation	p-values
Pearson	0.315	2.2e-16
Kendall	0.247	2.2e-16
Spearman	0.358	2.2e-16

A weak statistically significant linear correlation between the two variables was observed. However, it should be noted that these tests test only for linear relationships. The MI (mutual information) statistic from information theory is calculated, to test for non-linear relations.

Mutual information feature selection

The MI of two random variables measures the mutual dependence between them (Cover & Thomas, 2012). In simpler terms, MI quantifies the reduction in uncertainty about one variable given another variable. It is calculated as:

$$MI(X;Y) = \sum_{x,y} P_{XY}(x,y) \log \frac{P_{XY}(x,y)}{P_X(x)P_Y(y)} \tag{11}$$

Note that the mutual information of two independent variables is 0. When two variables are independent, the product of their marginal density functions, $P_X(x)$ and $P_Y(y)$ equals the joint probability density function $P_{XY}(x,y)$. Thus, the sum in (11) becomes zero. Unlike correlation, MI is able to measure any relationship between variables.

The meaning of the exact value of the MI is difficult to interpret. Several universally available variables from the ISDA database were tested on their MI with the usage variable. Those with a MI larger than zero were included in the usageIntensity prediction models. Binned combinations of *Market*, *MarketGroup* and *Repair* were found to have a MI larger than zero.

Inherent difference between systems with and without usage data

It is important to note that using data from systems with usage data to predict usage of systems without usage data implies an assumption. This assumption is that there is no inherent difference between systems with and without usage data. Multiple parties involved with the collection and analysis of MIS data have confirmed the validity of this assumption. It is generally believed that illogical for purchased systems to be sporadically used due to their costs. The following issues are the cause of a majority of the gaps/absence of usage data:

- data might be discarded at a subsidiary due to lack of storage data, prior to upload to the ISDA database
- machine failure
- (ignored) networking issues
- discontinuation/transitioning towards new replacement systems

7.2. Training dataset 'usageIntensity' approximation models

Several decisions regarding the training and validation dataset have been made, based on the previous data exploration and the assumption that a heavily used system is repaired more often. First, the models are only trained using data from modern systems with *SystemCode* = [redacted]. Second, a different method for calculating the usage variable was chosen. Let $Y_i = (Y_1, \dots, Y_N)$ be the total duration of all examinations recorded on the ith day, recorded for an individual system.

$$usageDuration = \frac{\sum_{i=1}^{N} Y_i}{t_N - t_1 - \sum_{i=1}^{N-1} Gap(i)} \quad (12)$$

This approximation of the usageIntensity of a system requires fewer assumptions and is more exact. It records the number of hours a systems is operating per day on average. The histogram of this new usage variable in shown in Figure 7.6. This histogram bears a closer resemblace to the histogram of the repair variable as shown in Figure 7.5.

[redacted]

Figure 7.6 – Histogram usageDuration: SystemCode=[redacted]

SystemCode was also excluded despite it being identified as a statistically significant predictor during the feature analysis. The objective is to obtain a model that can be applied for any system type. The training dataset contains only [redacted] Including usage data from all system types is not an option as [redacted].

7.3. Training 'usageIntensity' approximation models

We are interested in 'usage' on a system level. When remaining within the boundaries of non-parametric survival analysis, REDA was deemed the most appropriate method for this method, as events are linked to individual systems. To be precise, the technique used is the Nelson-Aalen estimator of the cumulative hazard function, as implemented in the REDA R package (Wang, Fu, & Yan, 2017). General machine learning methods appropriate for the data are considered as viable alternatives and used for benchmarking.

Two target variables, two models

The trained models will be used to provide covariates for the RCF prediction model. Two different survival analysis methods are used, namely the KME and CoxPH. These two models take covariates in different formats. The KME is a univariate method, with the only variable being time. Covariates can be used by drawing different survival curves for individual groups. The CoxPH method is best provided with continuous variables. For this reason, both a classification model and a regression model are trained.

7.4. Testing and evaluating 'usageIntensity' approximation models

This subsection will provide some background information on the best performing model, random forests. Some explanation is provided about the metrics and methods used for model comparison. Several models were trained using multiple general machine learning techniques. In order to expedite the process of modelling and analysis for usageIntensity approximation, the decision has been made make use of multiple existing R packages that have seen frequent use in academic papers. In order to limit the length of this report, only the results of the best performing models for each objective are discussed here, the random forest classifier and random forest regression.

The formalization of the random forest algorithm is generally accredited to Leo Breiman due to his work at the turn of the millennium. The predictive power of the ensemble algorithm in both classification and regression was demonstrated in a series of papers by Breiman (1996, 2001, 2004). An ensemble of multiple decision trees is 'grown' using bootstrap aggregation, also known as bagging. Individual trees are trained on a different sample and grown by randomly selecting a number of features to split upon at each node. The final model results from a culmination of all individual trees in a fashion reminiscent to majority voting when classification is the objective. Regression forest are generally implemented as an ensemble of regression trees. Data is split into leaves using binary trees and separate regressions are performed in order to minimize the error on each leaf, Instead of a majority vote used for classification, the average is taken to combine the predictions of individual trees.

Random forest models were grown using the popular *randomForest* R package (Liaw & Wiener, 2002) instead of implementing the algorithm from scratch, as the approximation of usage of systems is not the main objective of this paper. Unlike similarly performing general-purpose learning methods, the random forest algorithm does not require (additional) resampling methods to obtain an unbiased error estimate. This is provided in the form of the oob (out-of-bag) error estimate. Each individual tree is grown using a random bootstrap sample from the provided training dataset. Roughly, a third of the entries are excluded from the sample used to construct the tree. For a binary classifier, the oob error estimate is calculated as the proportion of times the model trained on the larger bootstrap sample correctly predicts the true

class of cases that make up the smaller bootstrap sample. Multiple researchers have provided evidence for the notion that this method provides an unbiased estimator of the error (Bylander, 2002; Mitchell, 2011).

However, in order to allow for unbiased and reliable comparison to models obtained by other techniques, 10-fold cross validation is applied. Random forest models are evaluated using both the oob.error, as well as its error rate on different folds. After comparing different learning algorithms and selecting random forest, the final model used in the RCF prediction model was trained using the full dataset available.

7.4.1. Binary classifier performance metrics used for model comparison

A binary classifier was trained to allow the separation of systems into heavily and lightly used systems. Heavily and lightly used systems are defined as systems that are in use more and less than the median respectively. A confusion matrix of the best performing model on a random test sample ($N = 131$), obtained through a 10% holdout split, is given below in Table 7.4. This model was a random forest model containing 400 trees, in which all three features were considered at each split.

Table 7.4 – Confusion matrix binary classifier usage prediction model

	Pred usage<median	**Pred usage>median**
True Usage<median	52	14
True Usage>median	14	51

From this confusion matrix, several performance metrics are derived in order to evaluate the performance of the model, as shown in Table 7.5. The hyperparameters of the model scoring the highest on these metrics was chosen to be used for the model, trained on all available data, incorporated in the RCF prediction algorithm. One notices a balanced distribution of the different cases. This is to be expected given the nature of the target variable.

Table 7.5 – Performance metrics binary classifier usage prediction model

Performance metric	**Formula**	**Value**
Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$	0.7863
True Positive Rate	$\frac{TP}{TP + FN}$	0.7879
Precision	$\frac{TP}{TP + FP}$	0.7879
True Negative Rate	$\frac{TN}{TN + FP}$	0.7846
False Positive Rate	$1 - TNR$	0.2154

The ROC (receiver operating characteristic) curve is obtained by plotting the true positive rate against the false positive rate at different threshold settings. The area under the ROC curve is another metric useful for comparing different models. Figure 7.7 below shows the ROC curve of the aforementioned model

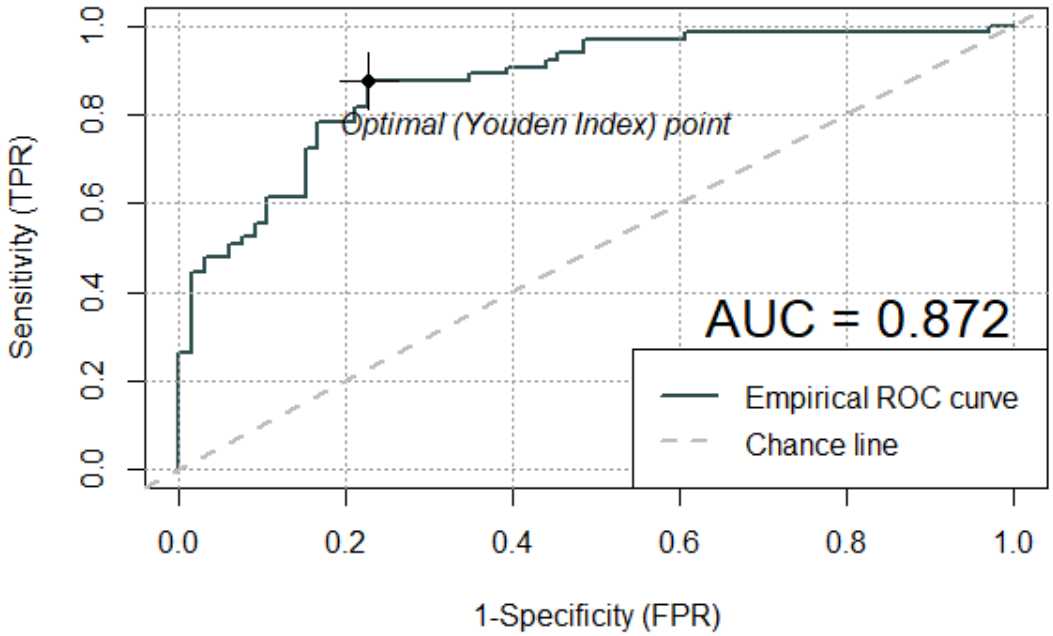


Figure 7.7 – ROC curve binary classifier usage prediction model

7.4.2. Regression performance metric, Mean Squared Error

As stated in Section 7.3, models were trained for a second objective, namely approximating the *usageIntensity* using regression. The average MSE (mean squared error) resulting from a 10-fold cross validation test was used to select the best performing model to be incorporated in the RCF prediction model. Models obtained using the random forest regression method resulted in the lowest MSE, on average.

The average MSE for both the 10 test sets and oob cases obtained are given in Table 7.6 below. The 'mtry' row indicates a hyperparameter of the random forest model, namely the number of features considered at each split.

Table 7.6 – oob.MSE and avg test.MSE from 10-fold cross-validation per 'mtry'

mtry	**1**	**2**	**3**
oob.err	1.12848	1.24298	1.35663
test.err	1.30345	1.44277	1.55321

Additional insights can be derived from the importance measures provided by the random forest algorithm. These measures indicate the importance of each feature. More specifically, it is calculated as the decrease in prediction accuracy on the oob sample when a feature is randomly changed/permuted e.g. its predictive power is removed.

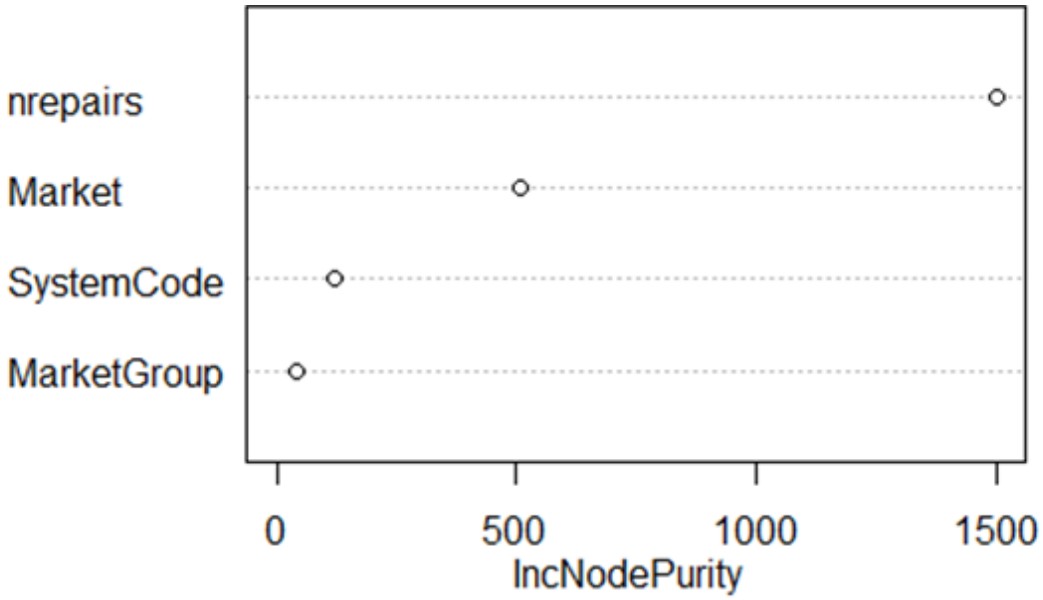


Figure 7.8 – Importance measures best usage regression forest model

The Gini-impurity is a performance metric commonly used for classification and regression trees. For classification trees, it is a measure of the probability of the tree placing an instance in the wrong leaf. It is calculated as follows: $Impurity_{Gini}(P) = 1 - \sum_{i=1}^{J} p_i^2$, where J is the number of classes $i \in \{1,2,\dots,J\}$ and P_i is the proportion of instances labelled with class i in the set. For random forests, the mean of all gini impurities of all trees is taken and normalized using the standard deviation of the differences. One observes that the *repairIndividual* variable has the highest importance, followed by the Market variable.

7.5. Conclusion

Several observations and results are summarized in this section with the second research question, and scientific validity in mind. It should be noted that the use of this model in the RCF prediction algorithm heavily relies on a not quite straightforward assumption, namely that the systems for which little to no usage data is available are similar to those for which data is faithfully recorded. It is important to consider that there might be a fundamental difference between systems with and without usage data. Assuming that the usage of data-poor systems can be predicted using data from other data-rich systems implies the assumption that the lack of data is not due to an inherent differences between the two. While this could not be conclusively ruled out in Section 7.1, several insights from experts on the have led to the decision to include the usage prediction model in the RCF prediction model.

Distinguishing parts subject to wear using survival analysis

Another point of interest indicated by Philips is whether survival analysis can be used to indicate whether a system component is subject to wear. This knowledge could be used to reduce the noise of failures of parts not subject to wear when predicting the usage intensities using corrective replacements. Should it be known which parts are subject to wear, it should

be possible to obtain a *repairIntensity* more strongly correlated to the *usageIntensity*, by using only corrective failures of parts subject to wear to calculate the *repairIntensity*. *RepairIntensity* per system would be calculated by averaging the number of corrective replacements of parts subject to wear per year instead of all parts, regardless of their failure distribution.

The KME is limited in its applicability for this objective. The KME in combination with the log-rank test could be used to detect statistically significant differences in survival rates over time among systems grouped on their *usageIntensities*. However, differences due to other factors such as the market or the version of the part cannot be distinguished. Limiting data to a single part version or systems from a single market might result in an inadequately sized dataset.

Thus, fitting CoxPH models is the better alternative when sufficient data for estimate convergence is available. The multivariate nature of the CoxPH models allows for the isolation of the effect of the *usageIntensity* covariate on the hazard rate/average time until failure. Fitting CoxPH models also allows for the quantification of the extent of the effect. The individual variable estimate $\beta_{usageIntesity}$ can be interpreted as follows:

Increasing the number of hours a system is in use per day on average by 1

results in an increase of $\beta_{usageIntesity}$ *in the hazard rate at time t*

Several metrics and statistics describing model fit of the CoxPH have been evaluated since 1972. Most frequently used are the likelihood ratio test and Wald test. Naturally, a p-values threshold is used to decide whether the observations are statistically significant. Model comparison is facilitated by the AIC (Akaike Information Criterion). As the CoxPH model deals with censored data, a modified version of the BIC (Bayesian Information Criterion) was proposed by Volinsky & Raftery (2000) for variable selection, where focus was shifted to uncensored events, e.g. observed events/mortalities. Should the feature space be expanded to a higher dimensionality, e.g. when a model is developed for a specific part such as the X-ray tube, there are better alternatives. Luo, Xu, & Chen, (2015) have shown the Extended BIC (J. Chen & Chen, 2008) to perform consistent in the CoxPH model.

No widely accepted equivalent of the R^2 statistic for the CoxPH model could be found at the time of writing. The R^2 values returned by the coxph() function from the 'survival' R package are also known as pseudo-R^2 values. These are not equivalent to the R^2 values returned in ordinary linear regression. Instead, the R^2 value denotes the improvement between a null-model and the fitted model. Squared residuals of the fitted model divided by the squared residuals of the null-model. These values are often quite low in CoxPH regressions, depending on the data. At least much lower than the usual >0.8 rule of thumb commonly used for assessing model fit of linear regression. A more frequently used metric for evaluation CoxPH models is the concordance index. Concordance is calculated as the proportion of pairs of cases for which the order of occurrence of the event correctly predicted.

Part IV – Modelling

8. Root-cause failure prediction model

In this section, the applicability of survival analysis methods for data driven root cause analysis is researched, through the conceptualisation of a RCF prediction model. The answer to the third research question will be derived from this and the subsequent sections. This section contains multiple intermediate results obtained throughout the project from deprecated models of varying levels of completeness. The iterative development process of the finished RCF prediction model is described in detail. In this section, the modelling decisions and the findings that they are based on, are discussed. These decisions include, but are not limited to:

- Tuning of hyper parameter settings (through grid search or other methods)
- Feature selection/engineering decisions
- Elementary model selection

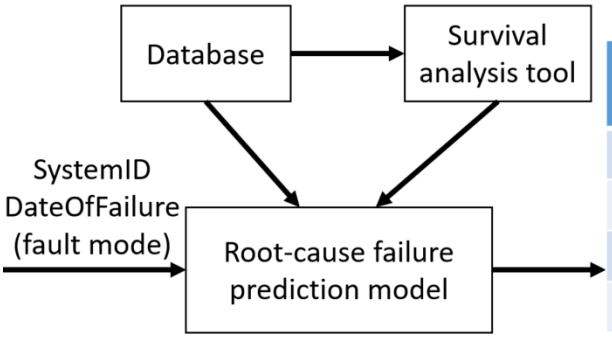
8.1. Methods

As mentioned in Section 3.4.2, this project is limited to non-parametric methods. One of the most extensively researched and popular survival analysis methods is the KME. The equally popular CoxPH model is investigated as a multi-variate alternative. An in-depth explanation on both the KME and CoxPH model can be found in Section 4.1.1 and 4.1.2 respectively. Both methods are implemented using the 'survival' R package (Therneau, 2015).

8.2. Model concept

The structure of the conceptual model is outlined in Figure 8.1 below. The model is separated into two parts. The secondary part consists of the survival analysis tool. This part is implemented as a stand-alone tool, ensuring maintainability and allowing it to be used for objectives other than RCA. The main part consists of the RCF prediction model itself.

A MIS is considered to consist of multiple non-identical parts. It is assumed a failure of any individual part will bring the system to a halt. Failure of the system is instantaneously noticed without inspection. The replacement of a single part is needed to repair the system. It is assumed that there is no dependence of any type between parts.



$Part$	t	$S(t)$	$\widehat{\Delta S(t)}$	$prob\{part = root\ cause\}$
a	21	0,9	−0,28	0,64945
b	66	0,6	−0,02	0,25641
c	34	0,75	−0,06	0,09413
d	257	0,999	0	0,00001

Figure 8.1 – Conceptual model root-cause failure prediction model

To illustrate the idea behind the RCF prediction algorithm, let us assume a hypothetical situation of complete information, e.g. the behaviour of parts in terms of failures is known to us. This behaviour can be expressed mathematically using definitions borrowed from reliability engineering theory (Ebeling, 2004), the focus of which is the following expression of reliability as a function of time:

Reliability $R(t)$ denotes the probability that the time of failure T of a part is greater than t

Consider a simple system consisting of three different parts $p \in P = \{1,2,3\}$, with corresponding reliability functions $R_p(t) = \Pr(T_P > t)$, where T_p is the time of failure of part p. The failure of any part p causes system failure. Given $R_p(t)$, we know failure probability density distribution $f_p(t) = -\frac{dR_p(t)}{dt}$ of part p.

Consider a system being started with completely new parts at time 0. System failure was detected for the first time at time t'. Let $p_{RCF} \in P$ be the unknown part that has failed at time t'.

As no system failure had been detected before time t', and given that failure of the system is noticed immediately and that the failure of any part p causes system failure, we can state:

$$\Pr(p = p_{RCF}) = \Pr\big(T_p = \min(T_1, T_2, T_3)\big) = \frac{f_p(t')}{R_p(t')} \div \sum_{i=1}^{P} \frac{f_i(t')}{R_i(t')} \quad (13)$$

Let σ be a permutation of P and $\mathfrak{S}_P$ be the set of all possible permutations of P. In this situation, where $R_p(t)$ is by assumption known to us, we can formalize the following optimization problem, where σ^* is the optimal permutation:

$$choose\ \sigma^* \in \mathfrak{S}_P \quad s.t.\ \Pr(\sigma(1) = p_{RCF}) \geq \Pr(\sigma(2) = p_{RCF}) \geq \Pr(\sigma(3) = p_{RCF}) \quad (14)$$

Given (13), we can now solve the optimization problem described in (14):

$$\sigma^* = \left\{ \sigma \in \mathfrak{S}_P \,\middle|\, \frac{f_{\sigma(1)}(t')}{R_{\sigma(1)}(t')} \geq \frac{f_{\sigma(2)}(t')}{R_{\sigma(2)}(t')} \geq \frac{f_{\sigma(3)}(t')}{R_{\sigma(3)}(t')} \right\}$$

8.2.1. Model input

The RCF prediction model is provided the *ConfigID*, *CallOpenDate* and optionally a fault mode description that limits the parts that should be investigated, corresponding to a call that is solvable by the replacement of a part, as input. For simplicity, the *SubSystem* variable described in Section 5.1 is taken as an abstraction of such a fault mode description. This abstraction relies on the assumption that the remote service engineer is able to narrow the possible causes down to the *SubSystem* level.

The RCF prediction model queries the database in order to obtain covariates such as the *SystemCode*, *MarketGroup* and *repairIntensity*. Using these covariates, the model trained in Section 7.4 is used to obtain the final covariate *usageIntensity*. The covariates used in this

project is limited to universally available covariates, e.g. covariates obtainable from maintenance data. Other covariates obtained from usage and inspection data could be incorporated in projects that are more focussed on certain system types or parts.

8.2.2. Algorithm

The inner workings of the RCF prediction model can be summarized in the following steps:

1. **Identifying the current parts in the system and their age**

The exact age can be determined for a few parts that have underwent replacement before, within the window of observation e.g. the replacement was recorded in the database. For most parts however, the exact age needs to be approximated. The majority of parts within a system will not have failed within the window of observation. It is only known that these parts have not failed during the time that the system has been observed. This time is used as an approximation of the age of parts for which no replacements are recorded in the database.

$$Age = \begin{cases} time\ in\ days\ since\ last\ replacement, & casesRecorded > 0 \\ time\ in\ days\ since\ start\ window\ of\ observation, & casesRecorded = 0 \end{cases} \quad (15)$$

A slightly different alternative is explored in Section 9.3.3. After obtaining the list of suspect parts, the RCF prediction model extends the survival analysis tool for each part in the system:

2. **Fitting CoxPH models**

In contrast to the KME, which is completely non-parametric, applying the CoxPH method first requires the inference of estimates $\boldsymbol{\beta}$. The popular 'survival' R package (Therneau, 2015) is used which applies the method as described by Andersen & Gill (1982) who proposed reforming the problem as a counting process and apply martingale theory to consistently obtain the estimates. Numerous authors have proposed their own alternatives/extensions to the partial likelihood method since its presentation by Cox in 1972. Tibshirani (1997) proposed a new method for variable selection using the LASSO regularisation method described by him in the previous year. Fan & Li, (2002) extended their Smoothly Clipped Absolute Deviation regularization method to the CoxPH model as an alternative to the LASSO method. Efron et al. (2004) proposed their Least Angle Regression solution, which was closely related to the LASSO solution. Wu (2012) extended this work, included the elastic net penalty. The results of each method, stepwise or otherwise, differs marginally, with different methods providing the lowest bias, or variance. Given the low dimensionality of the feature space for this problem, little benefit is expected from applying these more advanced regularisation methods for variable selection.

3. **Fit survival curves using either KME or fitted CoxPH model**

Survival curves are plotted for each part using the 'survival' R package. In a practical situation, survival curves would be fitted beforehand and stored. When applying the KME, decision variables include whether to group on one or multiple factors. When applying a CoxPH model,

one should consider which covariates to include. Possible grouping of features or covariates are described in detail in Section 6.3.

4. Calculate predictors using the obtained survival curves

After obtaining the survival curves through either the KME or a CoxPH model, one needs to consider how to use predict the likelihood of a specific component to be the RCF, using the obtained information. The most obvious predictor is using the survival rate $\hat{S}(t)$. However, several alternative predictors should be considered for several reasons.

Alternative predictors

One of the most important reasons is the presence of a valuable piece of information, namely the knowledge that the system was still working the day before. To illustrate the significance of this piece of information, the KME survival curve for a part is shown below in Figure 8.2. This part is suspected to have a decreasing hazard ratio, meaning it has a burn-in period and becomes more reliable over time.

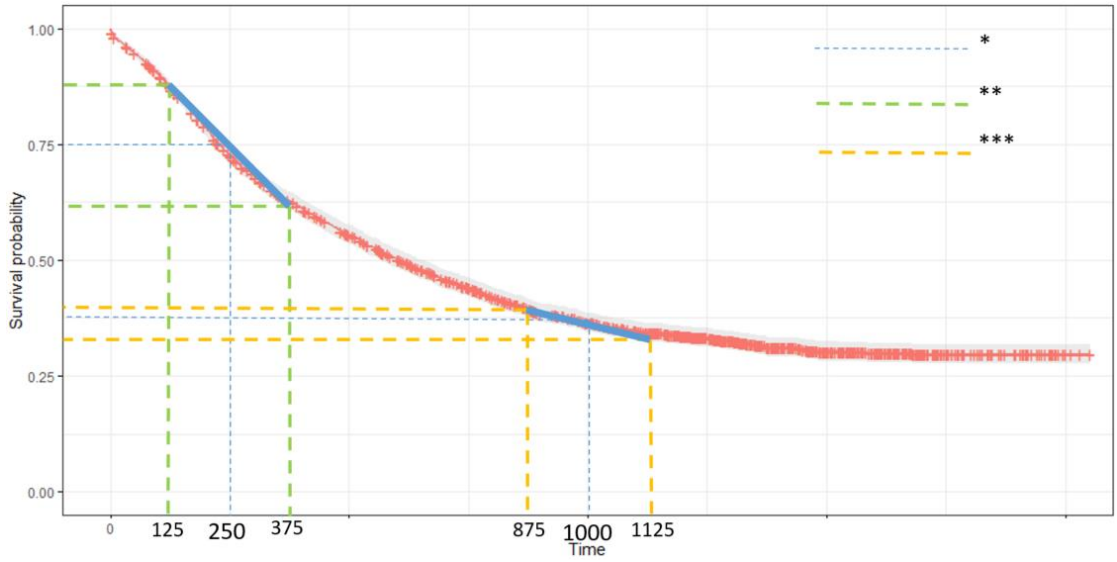


Figure 8.2 – Example KME, bayesian implications

One is able to observe that the curve is steeper around time 250 than around time 1000.

* The survival rate at time 250 is approximately 0.75.

* The survival rate at time 1000 is approximately 0.375.

** The $\hat{S}(t)$ drops from 0.875 to 0.625 between time 125 and 375 approximately.

*** The $\hat{S}(t)$ drops from 0.4 to approximately 0.3 between time 875 and 1125

A significantly higher number of parts per time unit fail around age 250 than around age 1000. Using the $\hat{S}(t)$ as a predictor would mean you would rank a part with an age of 1000 days to be more prone to failure than a part with an age of 250. Thus the hypothesis is that the change in $\hat{S}(t)$around t is a better predictor than $\hat{S}(t)$.

$$S(1000) = 0.375 < S(250) = 0.75$$

$$-\Delta\widehat{S(250)} > -\Delta\widehat{S(1000)}$$

$\widehat{\Delta S(t)}$ Predictor

There are several ways to approximate the change in survival rate predictor. One could use draw a line from the nearest changes in $S(t)$, or so called steps in the survival curve. However, this method would be quite subject to chance. The occurrence of two events at times t' and t'' with unusually short inter-arrival times could introduce a bias if the part in question has an age of $t\ where\ t' < t < t''$. Thus, the predictor is based on the change in $\hat{S}(t)$ during an adjustable time interval around the age of the part. For example, given a part that is 500 days old, the predictor is calculated as $\frac{S(450)-S(550)}{100}$, when the *DeltaRange* parameter defined in Section 8.2.4, is set to 50.

RA predictor

Another predictor derived from the survival curves is the RA predictor, defined as the remaining area under the survival curve, assuming the slope will remain the same. Taking the actual AUC (Area Under the Curve) is generally infeasible, except for only a handful of parts, due to the instability of the right tail of most survival curves obtained. To illustrate this, in Figure 8.2 one observes $\lim_{t\to\infty} S(t) \approx 0.25$, thus resulting in the true AUC being infinite. The RA predictor is derived from both $S(t)$ and $\widehat{\Delta S(t)}$ and calculated as follows:

$$RA(t) = \widehat{AUC} = \frac{1}{2} * S(t) * t = \frac{S(t)^2}{2 * \widehat{\Delta S(t)}} \tag{16}$$

The predictor can be seen as the relative likelihood a part will fail around an age of t days given that the part has survived for almost t days. It should be noted that the predictors are not probabilities e.g. a value between 0 and 1. This does not matter as we use the calculated alternative predictors, e.g. all predictors except S(t), for ranking solely. The actual probability of failure around age t does not matter, only its likelihood relative to the failure of another part around age t', where t' is the age of another component in the system at the *CallOpenDate*.

Frequencies of parts

The Log-transformed frequencies of occurrence of replacements are included in the calculation of predictors. As explained in Section 1.2, for an uncensored event to be recorded within a system, at least two consecutive replacements need to be recorded. This is seldom the case for highly reliable parts.

Confidence interval predictor

The difference between the confidence intervals was also investigated, but only after the introduction of a meta-learner as described in Section 8.3.1, as this predictor cannot be used alone to rank. This predictor is calculated as:

$$Conf(t) = \frac{upperbound(t) - lowerbound(t)}{S(t)} \tag{17}$$

8.2.3. Model output

The output of the model is a tuple of parts indicated by their *Part12NCs,* ordered on their probability of being the part that should be replaced to resolve the call. For complex systems or fault modes involving many parts, this list could be truncated to display only the probable parts or those with predicted probabilities above a certain (adjustable) threshold. In a practical situation, this output could possibly be supplemented with the weight, volume, cost and stock availability in local warehouses close to the input system of each part. A flowchart detailing the RCF prediction model algorithm is given below in Figure 8.3.

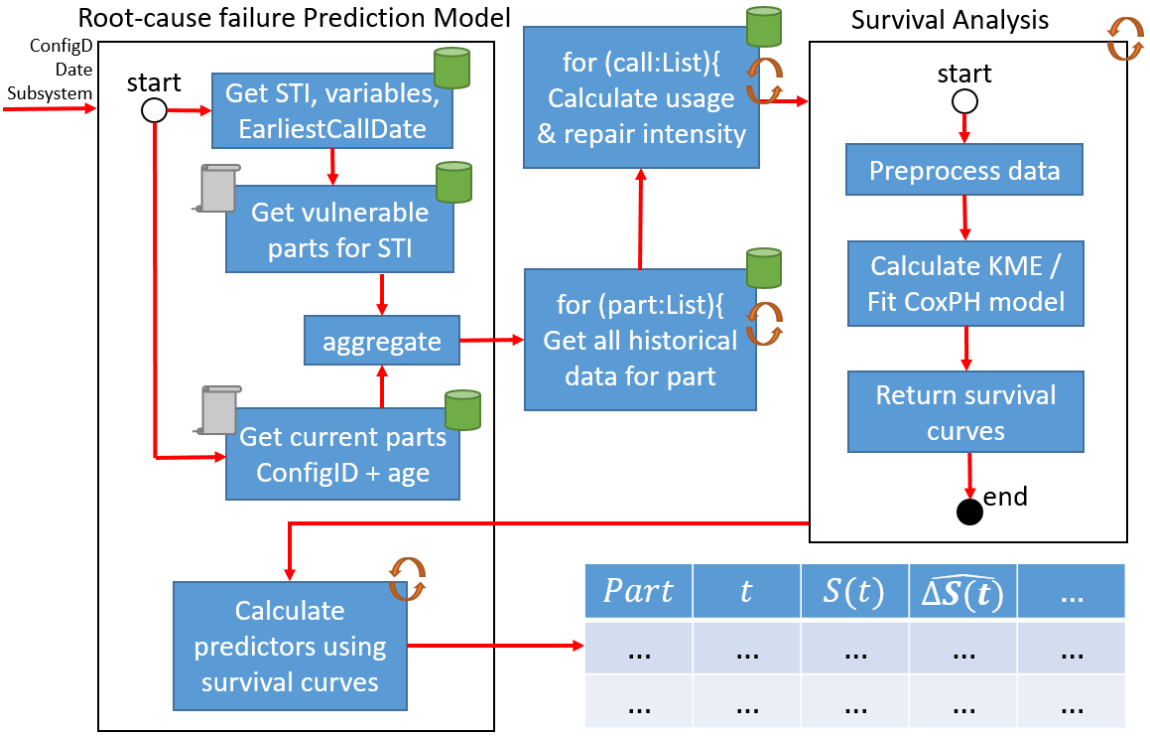


Figure 8.3 – Flowchart root-cause failure prediction algorithm

8.2.4. Model hyperparameters

The hyperparameters accepted by the prototype implementation are described in this section.

Retrospective length: τ

This hyperparameter describes the length of time τ in days. The model will only consider corrective replacements in the approximation of the usage covariate that are recorded before $CallOpenDate - \tau$. When passed this parameter, the model calculates the *repairIntensity* covariate of a system as a function of time. Let $Z_1, \dots, Z_N$ be a corrective replacement call, recorded for an individual system sorted on $CallOpenDate$. Let $t_1, \dots, t_N$ be the corresponding $CallOpenDates$.

$$RepairIntensity(t) = \begin{cases} \dfrac{\sum_{t-\tau \le i : t_i \le t}(Z_i)}{\tau}, & t - \tau \ge 0 \\ \dfrac{\sum_{i : t_i \le t}(Z_i)}{t}, & t - \tau < 0 \end{cases} \quad (18)$$

This hyperparameter was included under the assumption that both the usage and repair intensity of a system changes over time. Systems might be used more intensely over different periods. Depending on the erraticness of the usage/repair intensity, a shorter τ might be necessary to accurately gauge the covariate. However due to the relatively low occurrence rate of corrective replacements over time, this τ should be sufficiently large to reduce the influence of random chance. Time series smoothing methods such as exponential smoothing might be necessary when only considering a subset of part that are subject to wear. One observes in Table 8.1 below that the lowest average scores are obtained for the C-arm stand subsystem in systems with the *SystemCode* = 718094 are obtained when τ = 365, e.g. 1 year.

Table 8.1 – Average scores per τ, 718094 C-arm stand

τ	S(t)	S(t)/ln(freq)	ΔS(t)	ΔS(t)*ln(freq)	RA	Benchmark
730	14.06667	14.28889	14.44444	14.46667	14.46667	14.73333
183	12.88889	11.44444	12.71111	12.22222	12.71111	14.73333
365	12.06667	11.44444	12.4	12.15556	12.48889	14.73333

However, from Section 0 onwards, τ is set to $Max(CallOpenDate) - t$, in order to reduce the training time, e.g. the '*repairRatio'* variable as described in Section 7.1.1 is used.

Slope approximation interval: DeltaRange

This hyperparameter determines how the slope of a survival curve in approximated. Given a part with an age t at the date of system failure, the $\widehat{\Delta S(t)}$ predictor is defined as follows:

$$\widehat{\Delta S(t)} = \begin{cases} \dfrac{S(t - DeltaRange) - S(t + DeltaRange)}{2 * DeltaRange}, & t - DeltaRange \geq 0 \\ \dfrac{S(0) - S(t + DeltaRange)}{t + DeltaRange}, & t - DeltaRange < 0 \end{cases} \quad (19)$$

The reasoning for including this parameter is the same as the previous one. A longer period results in a less accurate prediction when the slope is erratic. Observing a shorter period is more subject to random chance.

It is worth mentioning that splines would be a viable alternative to approximate the slope. One would use the AIC and BIC to determine the right fit. However, the use of splines to smooth empirical curves is generally limited to improve the visual clarity rather than improve the accuracy of predictive algorithms. Introducing splines means introducing several inaccuracies and issues that are to be considered. In addition, in order to limit the length of the project, the decision has been made to limit failure distribution techniques to non-parametric methods. Thus, splines were not considered during this project.

A rough test was conducted to select an optimal *DeltaRange* parameter setting. The average score obtained, when ranking on the $\Delta S(t) * \ln(freq_i)$ predictor, on a test dataset of corrective replacement calls, are plotted in Figure 8.4 below. The first dataset contained

corrective replacement calls involving a part within the 'C-arm stand' subsystem of systems with a *SystemCode* = 718094. The second dataset contained corrective replacement calls involving a part within the 'UI modules' subsystem. The lowest scores were obtained when *DeltaRange* was set to approximately 125 days. Thus for the following experiments reported in Section 9, the *DeltaRange* parameter is always set to 125. This means that the slope is approximated over a period of between, 125 and 250 days, centered on the age of the part.

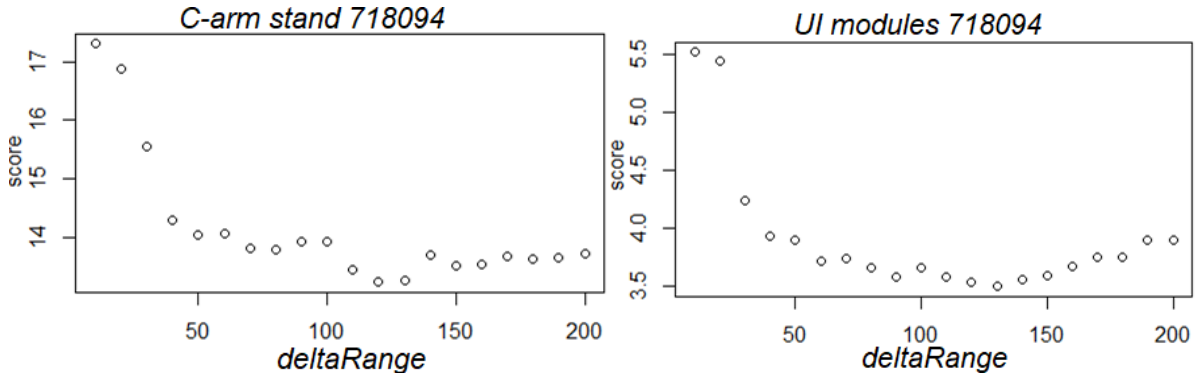


Figure 8.4 – Average score by DeltaRange: 718094 C-arm stand + UI modules

Cox Proportional Hazards or Kaplan Meier Estimator

Models build upon the KME consistently outperformed models build using predictors obtained from CoxPH models during the development of the algorithm. Model performance equalized when the usage prediction model was included in the algorithm to obtain approximations of the usage of individual systems. To illustrate this, some intermediate results are shown in Table 8.2 and Table 8.3 below. Modelling decisions were made iteratively, as each new model was evaluated using the same test dataset **including only systems younger than 2 years**. The average score of the $\frac{S(t)}{ln(freq)}$ and $\Delta S(t) * \ln(freq)$ predictors obtained using both the KME and CoxPH methods are shown in the 3rd and 4th column. The score of a call is equal to the rank of the true RCF as described in Section 9.1.1. The lower the score, the better.

Table 8.2 – Intermediate results: algorithm development, SystemCode=718094, SubSystem=C-arm Stand

Method	Covariates/grouping variable	S(t)/ln(freq)	ΔS(t)*ln(freq)	Benchmark
KME	MarketGroup	12,02222	17,97778	14,73333
CoxPH	MarketGroup + Part12NC	15,57143	17,40476	14,73333
KME	repairIntensity (REDA)	11,44444	12,15556	14,73333
CoxPH	repairInt. (REDA) + MarketGroup + Part12NC	11,53333	14,46667	14,73333
KME	usageIntensity	11,68889	12,57778	14,73333
CoxPH	repairIntensity + MarketGroup + Part12NC + usageIntensity	11,24444	12,22222	14,73333

Table 8.3 – Intermediate results: algorithm development, SystemCode=722006, SubSystem=Clea

Method	Covariates/grouping variable	S(t)/ln(freq)	ΔS(t)*ln(freq)	Benchmark
KME	MarketGroup	10,88636	11,47727	9,795455
CoxPH	MarketGroup + Part12NC	11,40909	12,15909	9,795455
KME	repairIntensity	8,181818	11,25	9,795455
CoxPH	repairIntensity + MarketGroup + Part12NC + usageIntensity	8,522727	10,88636	9,795455

One observes that models based on the KME initially outperform their CoxPH counterparts when the feature space in limited to only *MarketGroup* and *Part12NC*. Performance of the CoxPH models increases as more features are added. Comparison of the 3rd and 5th row of Table 8.2 shows that grouping on the *repairIntensity* variable, rather than the *usageIntensity* variable results in the better performing KME-based RCF prediction model, despite the usage prediction model including *repairIntensity* as a feature. It is difficult to draw conclusions on which is the better grouping variable. There are numerous uncertainties surrounding the *usageIntensity* variable as it is predicted using a supporting model, rather than measured empirically from the data. There simply is a significant probability of the used *usageIntensity* variable being highly biased and inaccurate, due to it being trained on data from other system types, rather than the one in question.

8.3. Extending the algorithm, appending a meta-learner

The final step of algorithm is to simply rank the parts on a given predictor. The KME $\widehat{S(t)}$ attempts to approximate the $R(t)$ from reliability engineering theory. Let N be the number of uncensored observations used to calculate the KME. Assuming that the assumptions made are correct, we can state that $\widehat{S(t)}$ approaches $R(t)$ as N approaches infinity.

$$\lim_{N\to\infty} \widehat{S(t)} = R(t) \wedge \widehat{\Delta S(t)} = f(t) \tag{20}$$

Given the solution to the defined optimization problem (14) in the hypothetical situation described in Section 8.2, we would expect $\widehat{\Delta S(t)} \div \widehat{S(t)}$ (hazard rate) to be the best predictor in a situation of high data availability. This was not found to be the case, meaning that we lack sufficient data to closely approximate the failure behaviour $R(t)$, made inaccurate assumptions or a combination of both. Unknowns in the data gathering process forced us to resort to rough assumptions in order to approximate the times of failure and the ages of parts. N did not exceed 3000, even for the most data rich fault mode explored in Sections 9 and 10. When the age of systems was not restricted, the KME or CoxPH alone were not enough to outperform the benchmark, given the data available. Thus, machine learning in the form of a meta-learner was applied in order to make the algorithm more robust to noise and scarcity in the data.

Intermediate results showed that while some predictors perform better on average, there is not one predictor that performs the best for every call. Furthermore, predictors based on the KME outperform predictors based on CoxPH models for some calls, and for vice versa. To illustrate this, Table 8.4 below shows the scores of several predictors based on the CoxPH model for a handful of cases. One observes several calls that are predicted more accurately by the $SQ = \frac{S(t)}{ln(freq)}$ predictor and other calls that are ranked higher by the $De = \Delta S(t) * \ln(freq)$ predictor.

Table 8.4 – Intermediate results: average score per predictor for test data

ConfigId	CallOpenDa	Part12NC	MaterialDescription	scoreS	scoreSQ	scoreD	scoreDe	scoreAUC
[redacted]				5	15	16	16	15
				7	14	24	24	23
				14	2	11	9	11
				44	18	42	42	42
				44	18	43	43	43
				6	4	2	2	3
				11	8	14	11	13
				13	9	21	20	21
				6	4	12	11	12
				8	1	4	4	4
				35	10	26	24	26
				18	32	3	4	3
				37	1	2	2	2
				12	10	11	10	11
				6	4	11	10	11
				5	4	2	3	2
				31	9	34	33	34
				39	8	39	39	39
				15	2	10	8	10
				7	4	4	3	4
				9	1	7	5	7
				6	15	18	17	18
				5	13	5	4	5
				21	28	21	22	20
				36	40	26	29	26
				31	16	11	10	13
				1	7	2	3	2
				5	4	5	4	4
				6	20	5	7	5

Table 8.5 below shows that ranking on a single predictor was outperformed by the benchmark, when the age of systems was not restricted. The first four columns present the scores obtained when solely using one predictor from different survival methods. The final column shows the scores, as described in Sections 9.1.1 and 9.1.2, obtained using both predictors obtained from CoxPH regression. Section 9.3.3 explains how this score was obtained. A significant improvement in model performance was observed after adding a meta-learner.

Table 8.5 – Intermediate results: improvement achieved by appending a meta-learner, Brilliance CT 64 Channel

Algorithm	$\boldsymbol{Cox + S(t)}$	$\boldsymbol{Cox + \frac{\widehat{\Delta S(t)}}{S(t)}}$	$\boldsymbol{KME + S(t)}$	$\boldsymbol{KME + \widehat{\Delta S(t)}}$	**Benchmark**	$\boldsymbol{Cox +}$ **Meta-learner**
RankingScore	24,97	26,26	24,37	26,35	18,73	14.09
Not in top 5	0,905	0,920	0,914	0,919	0,792	0.657

8.3.1. Multi-level stacking ensemble

The observations presented in Section 8.3 have led to the decision to extend the algorithm by appending a meta-learner that uses the predictors derived from the survival curves as features. Using the principles of ensemble machine learning, a multi-level stacking method is applied to combine the strength of multiple heterogeneous models into a single multi-stage model. A highly aggregated overview of the ensemble framework is depicted in Figure 8.5 below.

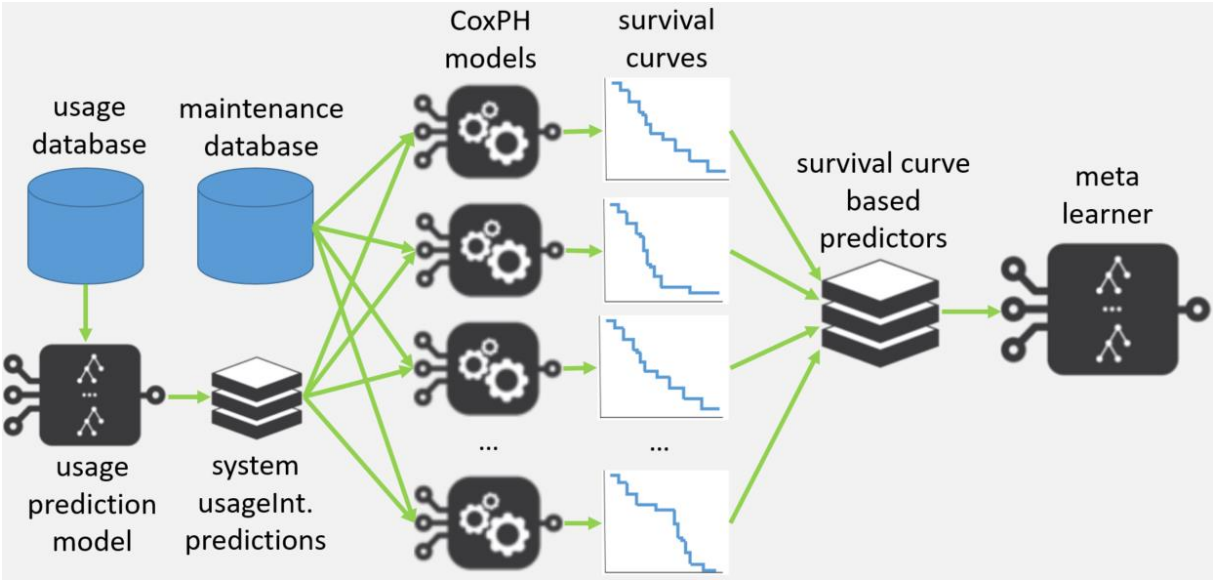


Figure 8.5 – Multi-level stacking ensemble framework

Each output prediction is based on the previous layer of predictions obtained from supporting models. There are several reasons to provide the meta-learner with the survival curves-based predictors obtained from the CoxPH models, instead of providing the training data directly:

- Enriching the model with domain knowledge and human intuition, preemptively performing some data manipulation, that is difficult to discover for machines
- Greatly reducing dimensionality of the feature space. Instead of providing all historical calls involving *n* parts belonging to thousands of systems as features for a single call, relations between certain historical calls are already obtained and provided in the form of survival curves for *n* parts.

8.3.2. Meta-learner/blender model

After settling on the stacking ensemble framework, a suitable meta-learner needs to be selected. The first candidate that might come to mind for ranking is 'learning to rank' algorithm. These algorithms have been extensively researched, often in pursuit of a backbone for search engines and document retrieval. However, one defining characteristic of these algorithms make them unsuitable for our objective. These algorithms attempt to rank all possible classes, whereas we are interested only in the rank of a single true class. For example, given a call for which the MIDAS board is the RCF, we do not care if the HLC-unit is ranked higher than the Connector Panel.

First option: Random forest

Thus, other options for the meta-learner are considered. Given the objective of predicting the right part to be the RCF, a supervised multi-class classifier is most suitable. Random forests have proven themselves powerful options both within and outside of this project. Their nature of combining predictions from multiple weak learners through majority voting provides a natural ranking system, namely ranking the classes by the number of votes received. Numerous other methods that satisfy these criteria exist, but one must start somewhere. The short training time and robustness of the random forest algorithm makes it as good a starting point as any other.

In order to evaluate whether the random forest is an appropriate meta-learner, a small test is executed. In order to expedite the development process, the least complex subsystem of the most numerous system type is chosen as a test case. This is the UI-module subsystem of the Pulsera BV systems. The ISDA database contains 3260 historical corrective replacement calls involving parts within that subsystem. 3086 calls remain after excluding calls within markets not present in the training data of the usage prediction model from Section 7. 286 calls are randomly sampled, to be used as a test data set. This data is not used in training or validation. Parts other than the HandSwitch, FoodSwitch and Viewpad were grouped together as 'Other'. Figure 8.6 below plots the oob error rate as a function of the number of trees included in the model.

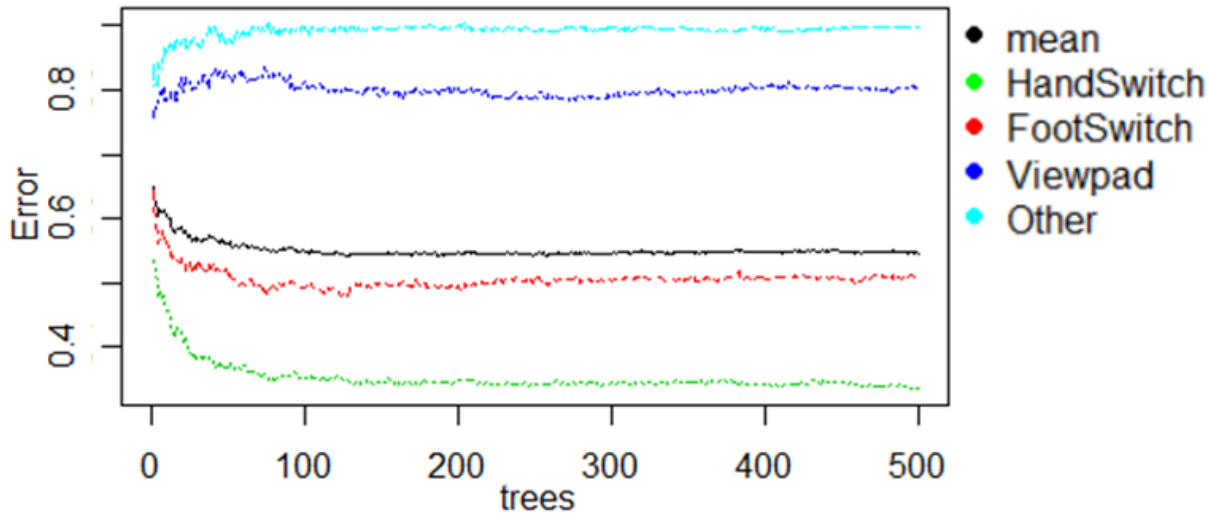


Figure 8.6 – 4-class random forest error rate

A known weakness of classic random forests can be observed. Namely that model performance suffers when attempting to predict underrepresented classes in unbalanced datasets (Chen, Liaw & Breiman, 2004). One observes that the model focusses on improving its ability to predict the two most numerous cases, namely the hand and footswitch. Without any weighing techniques, resampling methods or algorithm alterations, the random forest algorithm opts to focus on predicting the two most numerous classes as it is the most efficient way of optimizing the total average oob.error, represented by the black line.

Resampling methods

There are several strategies commonly used to deal with unbalanced datasets. Several resampling techniques are explored first. One of the most popular methods is the SMOTE (Synthetic Minority Over-sampling Technique) resampling technique (Chawla et al., 2002). Instead of simply duplicating (increasing weight of) cases, one approximates the distribution of the given data when applying SMOTE. In addition to under-sampling the minority classes, the minority classes are over-sampled by approximating the distribution and synthesizing new cases. Simpler resampling methods such as under- and over-sampling are explored as well. The results are summarized in Table 8.6 below. The error rates indicate the proportion of cases for which the prediction model correctly predicts the class, e.g. ranks the true RCF highest. The 'Ranking score' metric is described in Section 9.1, and obtained from the model predictions for the test dataset.

Table 8.6 – Comparison resampling methods

Sampling method	OOB err	Test err	Ranking score	Benchmark score
None	51.79%	53.15%	1.969	2.115
SMOTE	22.11%	58.04%	1.979	2.115
Over/up	24.44%	58.39%	2.031	2.115
Under/down	61.51%	61.89%	2.136	2.115

The SMOTE resampling method outperformed the other resampling methods in the oob error rate, test error rate and ranking score. Quite notably, the oob.error rate increased after applying under-sampling, indicating significant information loss. This model performed worse in terms of ranking the test dataset, being outperformed by the benchmark method. None of the models trained on resampled data performed better on the test dataset than the model trained on empirical data.

There is also the question of which metrics to use to evaluate model fit. Each metric has their own strength and weakness. The test error rate accurately measures the probability of the model predicting the true RCF when given a random instance from actual data. This metric however does not take into account the difficulty of predicting underrepresented classes. To give an example, a random forest model tuned to consider only two features at each split will (almost) never predict an instance to be of the Viewpad or Other class. Its test error rate however will be lower than a model that considers more than two features at each split (which occasionally predicts some instances to be of the Other or Viewpad class), due to the majority (approx. 69%) of the cases being of the Handswitch or Footswitch class. However, one can hardly argue a model that does not consider the possibility of an instance being of an underrepresented class to be a good fitting model.

Table 8.7 – oob.confusion matrix: 4-class random forest (Mtry=2)

		predicted				
		FS	HS	Other	VP	Class.error
True	FS	401	428	2	0	0.517
	HS	157	949	0	0	0.142
	Other	110	305	0	0	1
	VP	153	294	1	0	1

One could evaluate model fit using the error rate on weighted/resampled data samples. This is in fact a principle central to adaptive boosting. The model would be rewarded more heavily for correctly predicting instances of the underrepresented classes. The downside to this metric however is its distance to reality. The obtained metrics for over-sampled or synthesized data can't directly be compared to metrics for true datasets. Predicting these instances is 'easier' due to instances being more closely correlated. Some instances are duplicates (over-sampling) or synthesized from other cases (SMOTE).

No definite conclusion can be drawn from this experiment, other than SMOTE outperforming over- and under-sampling. The 'best fitting' model depends on the preferences and requirements of the user. Despite 'models trained on empiric data' outperforming 'models for which SMOTE is applied' on the test dataset, SMOTE should not be disregarded. The best fit depends on the requirements of the user. Applying SMOTE introduces additional issues to be considered. In order for the resampled distribution of data to resemble the true data distribution, the SMOTE technique requires enough data to fit a distribution, as well as the presence of an underlying distribution to detect.

Feature importance

Once again, advantage is taken from the bagging performed during the training of random forest models to obtain additional insights. Feature importance is depicted in Figure 8.7 below for the random forest meta-learner, where survival curve based predictors and system characteristics are provides as features. The confidence interval predictor was excluded as model performance suffered with the introduction of the predictor.

The single most important feature in a data poor environments to be the *SystemAge* feature, which is to be expected given what is known about the survival curves. As *SystemAge* passes a certain threshold, the features based on survival curves lose their predictive power. This notion was derived from the exploration of several survival curves of individual parts. The survival curves become less accurate as time approaches the right tail. As systems become older, the more parts will have survived long enough for their respective survival curve to become unstable at the age of the part. As survival curves become unstable, their predictive power suffers.

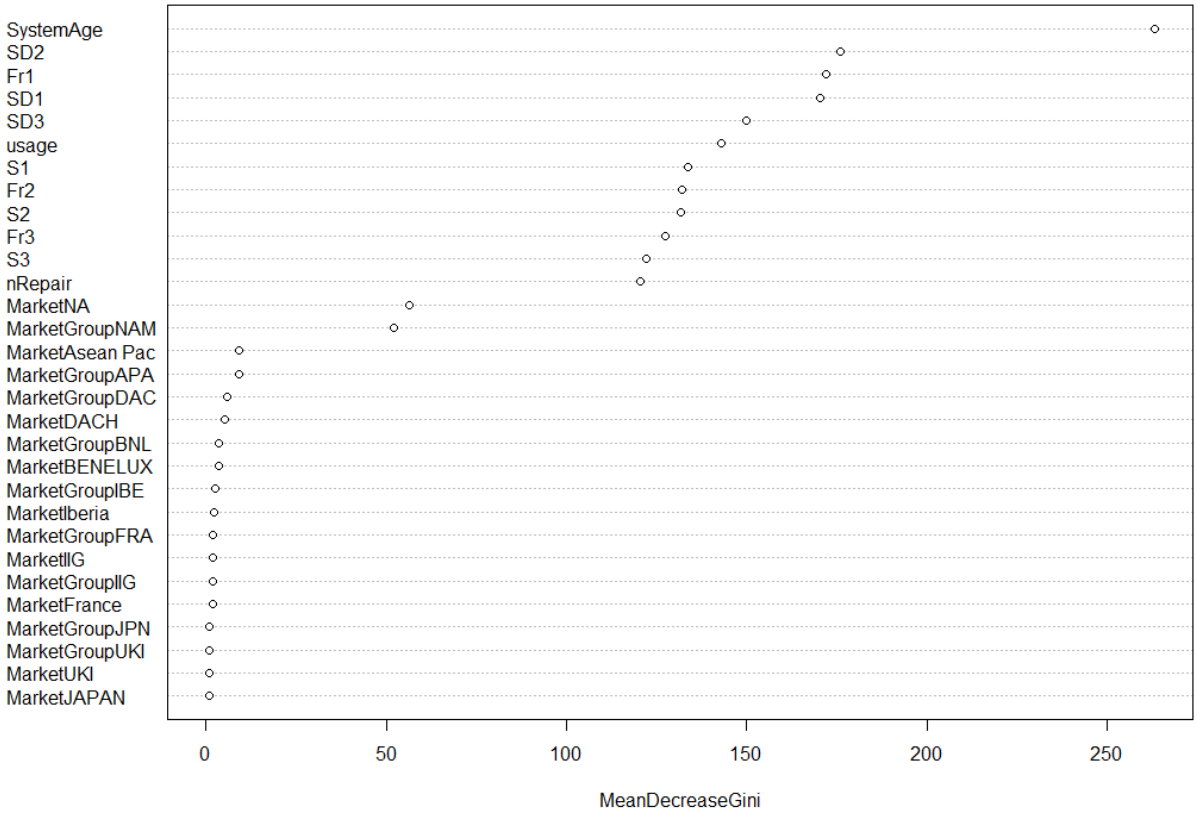


Figure 8.7 – Feature importance: 4-class random forest model

Random forest models are often referred to as black box models, meaning that their inner workings cannot easily be interpreted. It is however, possible to visualize individual trees in the ensemble. Part of the conditional inference tree with the highest accuracy in a conditional inference forest model (Hothorn, Hornik, & Zeileis, 2006) is depicted in Appendix E. One notices that its first step is to split all oob instances in 3 groups on the *SystemAge* variable. Survival curve predictors are mainly considered for the systems between the ages 574-2062 by this particular tree. It should be noted that there are still 499 other trees that influence the eventual predicted class for an instance. Exploring all trees through visual exploration is infeasible, which is why ensembles are classified as black box models despite consisting of multiple white box models.

As for the *Market* feature, there seems to be little importance in distinguishing between non-NA markets for the random forest model. Knowing whether the system is located in NA or not appears to provide relatively more information. This does fit common knowledge regarding the healthcare system in the United States of America, which is unique in its prevalence of private healthcare providers. The importance is presumed to be due to this market characteristic.

Second option: Boosting

Instead of applying resampling, it is also possible use algorithms that are inherently capable of dealing with unbalanced data as the meta-learner. Boosting is an umbrella term for methods that combine multiple weak learners iteratively into strong learners. These algorithms are inherently capable of handling unbalanced dataset as they iteratively add weak learners to the strong learner. AdaBoost (Adaptive boosting) for example increases the weights of wrongly predicted cases at each iteration. Thus weak learners created are focussed on increasing the strong learner performance in predicting the cases it is bad at predicting. The contribution weight of the new weak learner to the strong learner is scaled to its performance. Weak learners that perform greatly exert a greater change to the inner workings of the strong learner. The downside to this is that these methods do not handle noise well, as it attempts to fit every point, meaning it might get thrown off by noise data points.

Gradient boosting machines calculates the gradient of the loss function, with respect the prediction. Instead of changing the weight of the sample used to train the new weak learner, it trains weak learners on the remaining errors (residuals) of the strong learner. A gradient descent optimization process (Ruder, 2016) is used to determine the contribution weight of the new weak learner. Of these gradient boosting machines, XGBoost (T. Chen & Guestrin, 2016) is one of the most popular algorithms for machine learning competitions, praised for its performance and speed. Tables 8.7-8.9 below illustrate the comparison of the XGBoost algorithm and the random forest model with and without SMOTE resampling. Optimal hyper parameters settings were approximated using grid search.

Table 8.8 – test.confusion matrix: random forest (Mtry = 4)

		predicted				
		FS	HS	Other	VP	Class.error
True	FS	41	37	0	0	0,474
	HS	19	87	4	0	0,209
	Other	13	28	3	1	0,933
	VP	21	31	0	1	0,981

Table 8.9 – test.confusion matrix: random forest (SMOTE, Mtry = 8)

		predicted				
		FS	HS	Other	VP	Class.error
True	FS	32	21	18	7	0,590
	HS	19	50	40	1	0,545
	Other	6	4	29	6	0,356
	VP	14	20	15	4	0,925

Table 8.10 – test.confusion matrix: XGBoost (rounds=100,max-depth=10,colsample_bytree=0.5,eta=0.1,gamma=0)

		predicted				
		FS	HS	Other	VP	Class.error
True	FS	36	29	3	10	0,538
	HS	25	75	7	3	0,318
	Other	15	19	10	1	0,778
	VP	23	21	3	6	0,887

Several observations can be deduced from the confusion matrices above. The classic random forest model has the best error rate (0.539) and ranking score (1.895) of all models on the test set. However, one observes that the model achieves this by heavily focussing on the HS and FS classes and rarely predicting an instance to be of the minority class.

When one values the correct prediction of each class evenly, model performance on the test set drops drastically after the application of SMOTE. The error rate of the 'other' class decreased drastically, at the cost of a steep increase in the total error rate (0.598) and ranking score (2.112).

The XGBoost model performed better at predicting the under-represented classes while losing predictive power for the over-represented classes to a lesser degree. With an error rate of (0.556) and a ranking score of (1.972) on the test set, the XGBoost model presents a medium between to two alternatives. The model performs reasonable well on the true unbalanced dataset without completely writing off the under-represented classes. To recap, boosting methods focus on iteratively improving the performance of a strong learner, by adding weak learners that focus on the cases the strong learner is currently bad at predicting. The performance of an XGBoost model was found to be comparable to the RF models.

Third option: One-Vs-All binary classification

Given the objective of the RCF prediction model, applying a supervised multi-class classifier as the meta-learner is the obvious option. Nevertheless, the applicability of existing binary classifiers such as logistic regression and SVMs (Support Vector Machine) should not be ruled out. Several valuable performance metrics such as the ROC curve are limited to binary classifiers. Binary classifiers are frequently applied to multi-class problems using the One-Vs-All approach.

Applying this approach does change how the output is to be used. Following the beaten path, a winner-takes-all approach is often used in order to apply SVMs for multi-class classification. When thinking in terms of practical use, service engineers are currently already supported by an array of tools for root cause analysis. The RCF prediction model does not always have to return a single hard answer, neither is it likely that the prediction of a single

model will determine the taken course of action. No model will fit every maintenance call. Certain bits of information could already be obtained during calls with the clients, changing the situation. The model just has to contribute additional insights against reasonable cost, and support the decision making of the user in combination with other tools. Thus, there is arguably value to be gained to observe intermediate result from the steps within the One-Vs-All approach.

Understanding the ROC curve

Two popular binary classifiers are compared to the random forest model, namely logistic regression and SVMs. As mentioned before, one of the most prominent performance metrics used for binary classifiers is the ROC curve. Given a list of predictions for the probability that an instance belongs to the 'positive class', the ROC curve plots the TPR and FPR for different thresholds on this probability. A point on the ROC curve depicts what the TPR and FPR would be for a given threshold. For example, if the threshold is set to 0.6, the model would group all instances with a predicted probability >0.6 to be of the positive class, and vice versa. This probability is derived differently for each algorithm. For logistic regression, it is very straightforwardly taking the y value as the probability. For random forest binary classifiers, this probability is expressed as the proportion of votes for the 'positive' class. SVMs probabilities are calculated as a normalization of the margins from the fitted hyperplane.

Comparing binary classifiers

As stated before, three popular binary classifiers were investigated. Two versions of the random forest model (with and without SMOTE resampling) and SVMs (linear or polynomial kernel) were explored. Several metrics were used to evaluate model fit. The AUC (Area Under the receiver operating Curve) was deemed as one of the most useful metrics, together with an accuracy measure. Due to the large degree of imbalanced of classes in the dataset, the traditional measure of accuracy $(TP + TN)/(TP + FP + TN + FN)$ was deemed unsuitable for properly evaluating the model fit. For example, a model simply predicting every instance to not be of the Viewpad class would have an accuracy of 82.8%. Instead the 'balanced accuracy' as described in Brodersen et al. (2010) was used. This metric is calculated as $(TPR + TNR)/2$. Furthermore, the threshold probability for labelling was not set to 0.5, but to the Youden index. This is the threshold for which the difference between the TPR and FPR is greatest, formally defined as: $J = \max_t (TruePositiveRate(t) + FalsePositiveRate(t) - 1)$. This threshold point t^* optimizes the models ability to differentiate when classes are weighed equally. Each model is trained using 10-fold CV, with 10 repeats. The optimal threshold (Youden index) is recorded for each fold and the average over all 10 folds. This process is repeated 10 times, and the Youden index used for the final model is the average of all 10 repeats. A summarized overview of the results is given below in Table 8.11.

Table 8.11 – Overview binary classifier comparison

Handswitch vs ALL			Footswitch vs ALL			Viewpad vs ALL		
model	**AUC**	**Acc.**	**model**	**AUC**	**Acc.**	**model**	**AUC**	**Acc.**
RF	0.757	0.691	**RF**	0.702	0.632	**RF**	0.609	0.537
RF (SMOTE)	0.750	0.707	**RF (SMOTE)**	0.692	0.691	**RF (SMOTE)**	0.613	0.557
Logit	0.773	0.709	**Logit**	0.687	0.619	**Logit**	0.673	0.676
SVM (linear)	0.766	0.712	**SVM (linear)**	0.608	0.581	**SVM (linear)**	0.581	0.542
SVM (poly)	0.768	0.722	**SVM (poly)**	0.614	0.574	**SVM (poly)**	0.507	0.478

The handswitch is the single most prominent class in the dataset, accounting for 1216 of the 3086 calls (39.4%). The AUC varied only slightly for different models, with the lowest being observed for the RF mode with SMOTE (0.750) and highest being observed for the Logit model. The highest accuracy and balanced accuracy at the Youden index was obtained by the SVM with polynomial kernel.

Table 8.12 – Binary classifier performance on Handswitch vs all (N=1216, 39.4%)

Model	**RF**	**RF (SMOTE)**	**Logit**	**SVM (linear)**	**SVM (poly)**
AUC	0.757	0.750	0.773	0.766	0.768
$Accuracy_{0.5}$	0.696	0.682	0.703	0.685	0.696
$Accuracy_{Youden}$	0.696	0.703	0.703	0.717	0.731
Balanced $Accuracy_{Youden}$	0.691	0.707	0.709	0.712	0.722

Performance between different models differed much more significantly when predicting the footswitch vs all. The AUC of the SVMs was significantly lower than those observed for the other models. All models scored similarly on the traditional accuracy metric. However, when considering the balanced accuracy metrics, the random forest models performed best, with the random forest model with SMOTE applied being clearly superior.

Table 8.13 – Binary classifier performance on Footswitch vs all (N=909, 29.5%)

model	**RF**	**RF (SMOTE)**	**Logit**	**SVM (linear)**	**SVM (poly)**
AUC	0.702	0.692	0.687	0.608	0.614
$Accuracy_{0.5}$	0.759	0.759	0.738	0.745	0.745
$Accuracy_{Youden}$	0.685	0.769	0.661	0.664	0.636
Balanced $Accuracy_{Youden}$	0.632	0.691	0.619	0.581	0.574

The logit model was clearly the superior model when attempting to predict the Viewpad vs all, outperforming all other models significantly on every metric except the traditional accuracy metric. The SVM with a polynomial kernel model was the worst performing model by a significant margin.

Table 8.14 – Binary classifier performance on Viewpad vs all (N=531, 17.2%)

model	RF	RF (SMOTE)	Logit	SVM (linear)	SVM (poly)
AUC	0.609	0.613	0.673	0.581	0.507
Accuracy$_{0.5}$	0.850	0.852	0.850	0.850	0.850
Accuracy$_{Youden}$	0.490	0.462	0.846	0.633	0.546
Balanced Accuracy$_{Youden}$	0.537	0.557	0.676	0.542	0.478

Conclusion One-Vs-All binary classification

No universally superior binary classifier was identified. For predicting the Handswitch vs all, the SVM model with a polynomial kernel was found to marginally outperform the other models. The performance of this model however dropped quickly as the isolated class became less prominent. This relatively complex model outperformed the other models when isolating a dominant class. The random forest model with SMOTE significantly outperformed other models when isolating the moderately prominent class was the objective. The performance of the Logit model was most consistent, performing significantly better than the alternatives when predicting the rarest tested class.

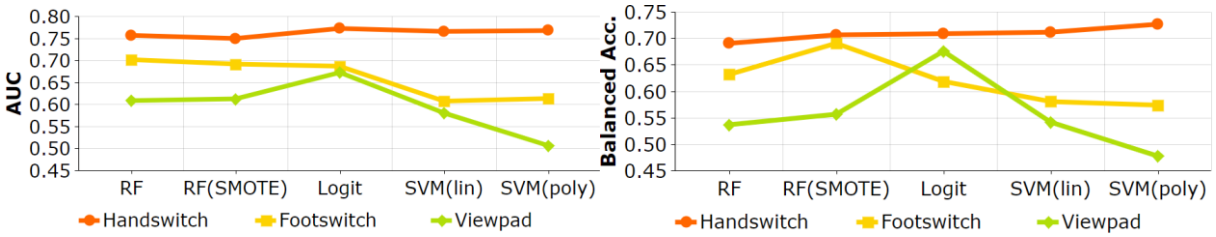


Figure 8.8 – AUC and balanced accuracy for binary classifiers

Each (sub)system will be different, including the parts with them. Van der Ploeg, Austin & Steyerberg (2014) have found in a simulation study that modern modelling techniques such as SVM, neural networks and RF may need over 10 times as many events per variable to achieve a stable AUC and a small optimism than classical modelling techniques such as the Logit model. It does not come as a surprise that as data availability for the isolated class becomes scarce, less complex and more robust models perform better. Given what is known about the scarce maintenance data availability for most parts, the application of SVMs will rarely be a good option. In most cases, the simple Logit model will be most suitable, which also excels in terms of training time and model interpretability. RF models are likely the best option when

data availability is moderate, e.g. for frequently failing parts within a subsystem such as the foot switch and geo module kit. Only for the most frequently failing parts such as the X-ray tube assemblies, will attempting more complex methods be warranted.

Part V – Results

In this part, the final implementation of the root-cause failure prediction model based on survival analysis is presented and its performance evaluated. The setting of parameters and elementary model selection were based on the results reported in the previous part. Possible practical applications of this method/model are discussed in more detail in the subsequent part. Given that the RCF prediction model targets several research gaps in existing literature, there is great uncertainty in what constitutes appropriate evaluation metrics. One can only speculate on whether the proposed methodology will be of any value in a practical setting.

Evaluation of the model performance was started with the most ideal circumstances, transitioning towards gradually more unfavourable demands and data availability. Should model perform insufficiently in the most favourable situation, there would be little reason to investigate its performance in the subsequent test environments.

9. Performance evaluation

In this section, the performance of the RCF prediction model is evaluated. The main objective of the model is to minimize the number of parts that are predicted to be more likely to be the RCF than the actual RCF. The results could be applied in practice to guide the service engineer in which part to diagnose next. In order to maximize the practical value generated by this project, the models are tested on real data where possible. Real maintenance log data was used to fill the training and test dataset.

9.1. Scoring & benchmarking

First there is a need to determine in which way to score the RCF prediction model. Two chosen methods are described in the following subsections. The scores of different models are benchmarked against a simple method where we rank parts solely on its historic frequency.

9.1.1. Average number of parts ranked higher than the RCF

The models are scored on the number of parts that it ranks higher (predicts to have a higher probability to be the root-cause) than the actual root-cause. Let $Rank_1, \dots, Rank_N$ be the rank of the true RCF, predicted by the model, for a call in the test dataset containing N calls.

$$Score_{avg} = \sum_{i=1}^{N} \frac{Rank_i}{N} \tag{21}$$

9.1.2. Proportion of cases the RCF is included in the top n parts

Taking the average ranking of the true RCF is an arbitrary scoring method that bears limited relevance to the true value obtained by the predictions. One could argue there to be little difference ranking the true RCF at the 50th or 100th as both predictions can be argued to be useless. Thus, another scoring method is added in an attempt to more closely approximate the number of useful predictions.

The second scoring metric is the percentage of calls in the test dataset in which the RCF was ranked in the top n. There is the complexity of the subsystem in question to consider when choosing n. For example, replacements have been recorded for a total of 336 different parts in the 'C-arm stand' subsystem of the systems with *SystemCode* = 718094 in the catmasterlist database. In contrast, replacements have been recorded for a total of 50 different parts in the 'UI modules' subsystem of the same system type. Ranking the RCF in the top 10 is a more difficult task for the first subsystem. Thus n is set as a percentage of the number of unique parts grouped in *ReplacementChains* in the subsystem.

$$Score_{top} = \sum_{i:Rank_i \leq n} \left(\frac{1^i}{N}\right) * 100\% \tag{22}$$

It should be noted that both, frankly arbitrary, metrics were chosen for their generalizability. No distinction is made between for example, a set of wheels and a MIDAS board. Identifying the former to be the RCF would be trivial, in addition to the cost of transportation being significantly lower. The volume, diagnosability, scarcity and cost of parts should be considered in future projects when focussing on one subsystem, system type or set of parts. Doing so here would be out of scope. This matter is discussed further in Section 12.2 below.

9.2. Training data set

Next is the specification of the training dataset used and the assumptions made.
Data instances included in the training dataset were subject to the following criteria:

- The calls entails a corrective replacement
- The calls are limited to parts within the subsystem in question

Data used was limited to the data recorded in the catmasterlist database, internally known as the catmasterlist, with an exception for obtaining the training dataset for the *usageIntensity* prediction random forest models. The reasoning for this decision was threefold. First, using data from a single Vertica table makes it easier to separate the data in training and validation datasets. The catmasterlist database is no longer in use, meaning no new entries are added daily. The need to aggregate duplicate data entries is also eliminated. This decision also makes the analysis a lot less computationally expensive, especially in regards to the cost of SQL queries, as material use per case in stored in a single table. And most importantly, the Teradata_OneEMS database lacks the 'subsystem' column.

Given a single maintenance call, the data used encompasses all historical calls from the catmasterlist maintenance database of all systems that share parts with the system in question. An example is given to make this description less abstract. Given the following corrective replacement call as input:

- CallOpenDate = ’12:07:53_2014-08-09’
- ConfigId = 556180
- SystemCode = 718094
- Subsystem = ‘UI modules’

The UI modules subsystem within systems with the *SystemCode* = 718094 contain only a handful of parts that fail frequently. However, this handful of parts are used in approximately 100 different system types. There are over 30 thousand individual systems that belong to one of these 100+ system types. For these 30,000+ individual systems, all historical corrective repairs/replacement calls are queried and used to calculate repair intensities.

Access to usage data is provided indirectly, allowing the RCF prediction model to call the model trained on true usage data and obtain predicted *usageIntensities*. A handful of markets, among which LATAM or GreaterChina, contained no systems with recorded usage. No prediction could be given for systems in these markets. Thus systems in these markets were excluded. Furthermore, both corrective and planned maintenance calls are provided as well as all this data is used in some part of the algorithm. The data is not limited to only replacements, but general repairs as well. Figure 9.1 below gives a rough overview of the data used by the algorithm.

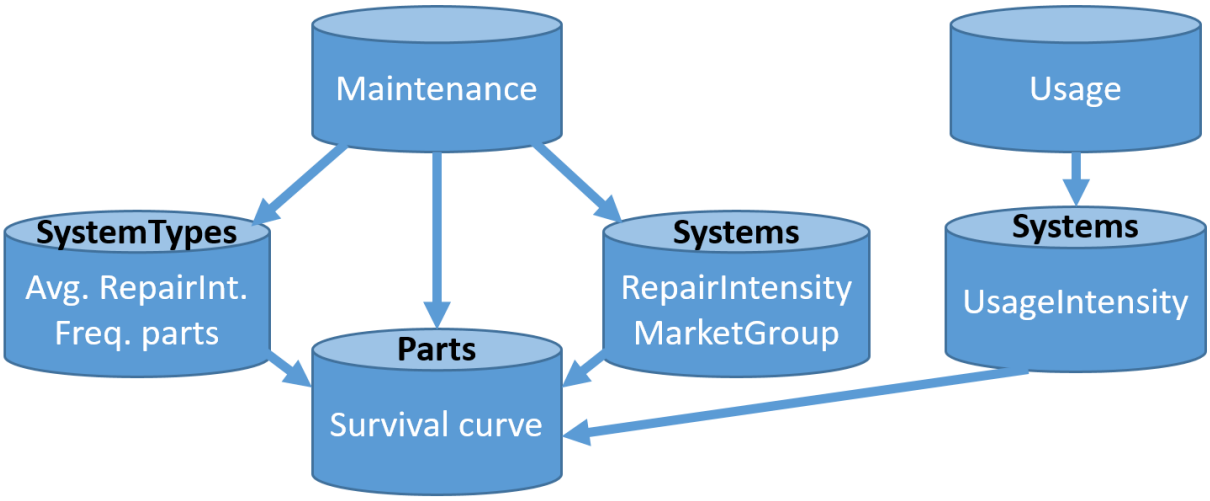


Figure 9.1 – Data used by the algorithm

9.2.1. ‘Future’ data

When researching the actual practical applicability of the algorithm, one could argue the need for consideration of whether or not to limit the available data to data entries that historically precede the test case. **In the previous sections, all data entries were included in the**

calculation of survival curves, including those that would be 'in the future at the time of the failure'. The absence of this restriction means that the analysis is not truly representative of a true situation.

9.2.2. Unnoticed parts

It should be noted that in this project only parts are analysed whose failure has been recorded before in the data. The researcher is unaware of all parts that are present in a given system or even a given subsystem. If the failure of a part has not been recorded before, it will have gone unnoticed during the analyses.

9.2.3. Excluding sporadically failing parts

A large portion of parts fail sporadically. For these parts, the time to failure cannot accurately be approximated with survival analysis or other methods due to lack of data. However, the fact that these parts rarely fail means that there is limited interest in these parts from a maintenance perspective anyway. To illustrate this, the distribution of historic occurrence of corrective replacements within the 'Clea' subsystem of systems with *SystemCode*=722006 is described. Replacements have been recorded for a total of 376 different parts, which can be aggregated into 290 different *ReplacementChains*. The replacements of the 40 most occurring *ReplacementChains* account for 89.834% of all historic replacements of parts within the subsystem. The remaining 250 *ReplacementChains* account for only 10% of all recorded replacement calls. The RCF prediction model is not evaluated on its ability to predict the failure of these rarely failing parts.

9.2.4. Multi label instances/calls involving >1 replacements

For some subsystems, a significant proportion (+10%) of calls involve more than one *Part12NC*. Some parts are nearly always replaced together, such as power units and connector plates. Multiple solutions to this issue are possible, the simplest of which is to exclude calls involving more than one part. In this project, one *Part12NC* was randomly selected whenever a call involved more than one part.

One could also change the problem from a multiclass problem to a multi-label problem. In multiclass problems, classes are mutually exclusive, meaning that an instance can belong to only one class. In multi-label problems, instances can belong to more than one class. An example would be predicting what genre a movie belongs to. A movie can belong to both the action and thriller genre. Translating this to the problem at hand, a call could involve both the collimator and viewpad. Different measure for accuracy would be used, such as the hamming-loss. One could also create additional classes for such instances, e.g. 'collimator-viewpad'. This would return a traditional multiclass problem. However, neither options were chosen due to time-constraints.

9.3. Results

9.3.1. *SystemCode*=718094 & *SubSystem*='UI modules'

The UI modules subsystem is one of the least complex, yet most data rich subsystems within the BV Pulsera systems. The top three most frequently failing parts account for approximately 85% of the historical corrective replacement calls. Thus, the second scoring metric will measure the percentage of calls in which the true RCF is ranked first. The error rate is equivalent to the proportion of parts not in the top (n=1) score metric.

Multi-class classifier Meta-learner

When solely considering the ranking score, the random forest model performs best. But this model was observed to rarely predict an instance to be of the underrepresented class. Thus this model can hardly be seen as a good fit. After the application of smote, the oob.error rate drops to ±0.22. But the performance on the test set drops drastically. The performance of the XGBoost model was close to the first model, without completely writing off the underrepresented classes.

Table 9.1 – UI modules multiclass classifier performance metrics

Model	**RF**	**RFSMOTE**	**XGBoost**	**BenchMark**
Ranking score	1.895	2.112	1.920	2.115
Error rate	0.539	0.598	0.556	0.615

Binary classifier One-Vs-All cascade

Instead of a meta-learner consisting of a single model, a cascade of binary classifiers was also evaluated as an alternative. This multistage meta-learner applies multiple, optionally heterogeneous, models sequentially. The same test dataset ($N = 286$) was used to evaluate the model ending with this meta-learner. The choices for the elementary models follow directly from the observations presented in Section 8.3.2. The cascade attempts to isolate classes in order of historic frequency. Table 9.2 below shows the intermediate results.

Table 9.2 – Binary classifier cascade UI modules: intermediate confusion matrices

		True
Pred.	Other	HS
Other	128	44
HS	33	81

Class: HS
Accuracy: 0.7308
Balanced: 0.7215
Sensitivity: 0.7950
Specificity: 0.6480

		True
Pred.	Other	FS
Other	99	31
FS	18	24

Class: FS
Accuracy: 0.7151
Balanced: 0.6413
Sensitivity: 0.8462
Specificity: 0.4364

		True
Pred.	Other	VP
Other	96	14
VP	9	11

Class: VP
Accuracy: 0.8231
Balanced: 0.6771
Sensitivity: 0.9143
Specificity: 0.4400

After the application of a SVM with polynomial kernel as the first binary classifier, 114 of the 286 instances were classified as ‘HS’, with 81 being correctly classified. 44 instances actually belonging to the ‘HS’ class were misidentified as not being of the ‘HS’ class. A random forest model with SMOTE was used to isolate instances belonging to the ‘FS’ class. 42 of the 172 remaining classes were predicted to belong to the ‘FS’ class, 24 of which correctly. Finally a logit model was applied on the remaining 130 instances. Only 20 instances were predicted to belong to the ‘VP’ class, of which 11 correctly.

The final confusion matrix is show below in Table 9.3. The total test.error rate of the cascade (0.493) is notably lower than the error rate observed for both the XGBoost (0.556) model and random forest model (0.539). Unfortunately drawing ranking scores from the cascade is not quite straight forward, though very feasible. Due to the number of possible options, each requiring several considerations, we focus on the class and total error rates for now. One also notes a disproportionately high number of instances (110 out of 286) being predicted as belong to the ‘Other’ class by the cascade meta-learner. The following distribution of classes was observed in the historical data:

HS = 39.4% FS = 29.5% VP = 17.2% Other = 13.9%

The cascade meta-learner predictions are distributed as:

HS = 39.9% FS = 14.7% VP = 7.0% Other = 38.5%

Table 9.3 – Binary classifier cascade UI modules: final confusion matrix

		predicted				
		FS	HS	Other	VP	Class.error
true	FS	24	18	28	3	0,671
	HS	1	81	39	4	0,352
	Other	8	7	29	2	0,370
	VP	9	8	14	11	0,738

Class: FS	Class: HS	Class: Other	Class: VP
Balanced: 0.6221	Balanced: 0.7215	Balanced: 0.6465	Balanced: 0.6125
Sensitivity: 0.3288	Sensitivity: 0.6480	Sensitivity: 0.6304	Sensitivity: 0.2619
Specificity: 0.9155	Specificity: 0.7950	Specificity: 0.6625	Specificity: 0.9631

After the application of the binary classifier cascade model, one is still left with 110 instances predicted to belong to the ‘Other’ class. This result can explained by the fact that the youden index is used as the threshold for predicting classes. The cascade model cherry picks the ‘easiest instances’/’instances it is able to differentiate best’. Only if an elementary binary classifier is very sure about an instance belonging to the ‘positive’ class, will it draw a conclusion and label it as such. If not, the binary classifier will simply push it off to the subsequent models. This leaves the impression that further improvement is possible. There

are several possible options that could be considered. The cascade model only draws a conclusion on the classes it is certain about. It is not the case that it is sure that the instances labelled 'Other' truly belong the class. For this reason, appending one of the aforementioned multi-class classifiers is explored in the next subsection below.

One might also attempt to optimize the threshold for labeling classes, depending on the wishes of the user. Naturally, one tries to maximize the TPR and minimize the FPR. Lowering the threshold for labeling classes as the 'positive' class will increase both the TPR and FPR. When creating a model for a practical decision support system, one needs to make a trade-off. However, domain knowledge is required for this. This option is not explored in this project due to the lack of generalizability.

Binary classifier One-Vs-All cascade + multiclass classifier

The cascade model is extended by using the multiclass classifier to predict the instances labelled as 'Other' by the cascade model. Appending the random forest model resulted in an error rate of (0.458). The confusion matrix is given below in Table 9.4. One notices that no instances were predicted to be of the VP class by the RF model and only one instance was left in the 'Other' class. 109 out of the 110 instances were predicted to HS or FS.

Table 9.4 – Binary classifier cascade + RF UI modules: confusion matrix

		predicted				
		FS	HS	Other	VP	Class.error
true	FS	34	36	0	3	0,534
	HS	10	110	1	4	0,120
	Other	14	30	0	2	1,000
	VP	13	18	0	11	0,738

Appending an XGBoost model resulted in a lower total error rate (0.444). The predictions were also more balanced with several instances labelled as belonging to the underrepresented classes. It outperformed the previous model in terms of class error rate for each class except the HS class.

Table 9.5 – Binary classifier cascade + XGBoost UI modules: confusion matrix

		predicted				
		FS	HS	Other	VP	Class.error
true	FS	39	28	2	4	0,466
	HS	14	98	8	5	0,216
	Other	13	21	9	3	0,804
	VP	13	13	3	13	0,690

Conclusion

It is difficult to state which model is the best. It basically depends on what the user prefers. The combined model has the highest performance of all models in terms of accuracy. But the separate binary classifier cascade might be more useful in practice. The penalty costs of mislabeling a corrective replacement call as 'Other'/'negative class' is arguably lower, as it could prompt a human engineer to look further into it. False positives might prove costlier as human actors might not be alerted and enabled to respond. The nature of the cascade model also makes it more valuable for decision support as some stages might consist of white box models. The Logit model for example sports a higher interpretability, increase the value and flexibility of the model for decision support.

9.3.2. *SystemCode*=718094 & *SubSystem*='C-arm stand'

A more complex subsystem investigated is the C-arm stand subsystem. Corrective replacement calls involving over 300 unique *Part12NCs* were recorded in the database. This problem was simplified to the top 20 *ReplacementChains* accounting for 70.7% of the 11000 calls recorded. The description of these parts and their frequencies in the dataset are given below in Table 9.6. A dataset of 5634 calls remained after several cleaning steps. Models were evaluated on the proportion of instances for which they ranked the true RCF in the top 2.

Table 9.6 – C-arm stand subsystem part list

Multiclass classifier Meta-learner

Two multiclass classifier models were evaluated on their ability to predict the true RCF for cases from the C-arm stand subsystem. Their performance based on the defined metrics are shown below in Table 9.7. The XGBoost model was found to have the best performance. The next observations are based only on the results from the XGBoost model.

Table 9.7 – C-arm stand multiclass classifier performance metrics

Model	RF	XGBoost	Benchmark
Ranking score	4.973	4.763	6.090
% RCF not ranked in top 2	0.667	0.603	0.706

Class-wise performance

Unsurprisingly, the RCF prediction model performed worse than the benchmark at classifying the 'Energy storage unit', which was the most frequent class in the dataset. The benchmark ranks this part as the most likely answer for every single instance. Figure 9.2 below shows the rank of the 'Energy storage unit' given by the RCF prediction model and the benchmark for all 93 instances in the test dataset where the true RCF was the 'Energy storage unit'. In the second image the instances are sorted on their score. The RCF prediction model deemed the 'Energy storage unit' mostly likely to be the RCF for 53 of the 91 instances (58.2%). The average rank of the 'Energy storage unit' predicted was 2.02 for these instances.

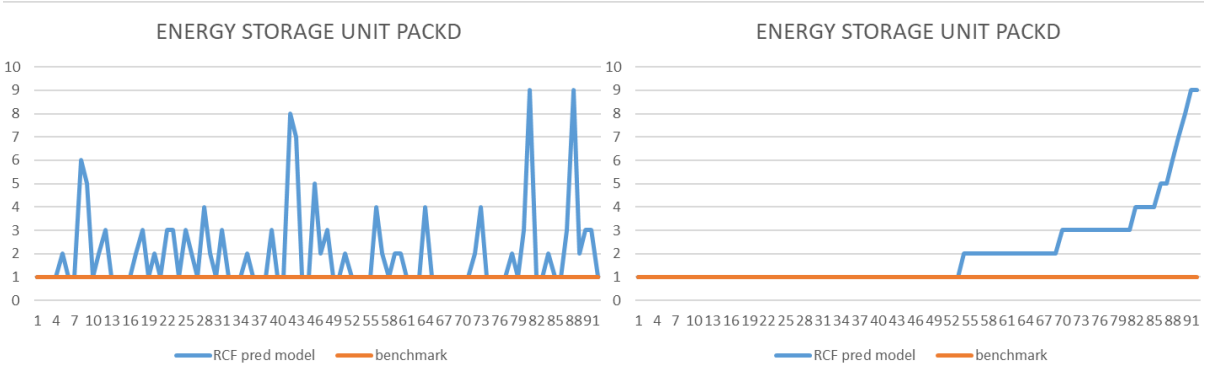


Figure 9.2 – C-arm stand: predicted rank of all 'Energy storage unit' instances

The classes for which the lowest class.error was observed were the 'IRIS + SHUTTER/ WEDGE UNIT FRU' (0.663) and 'C-ARC CABLE PULSERA' (0.566) classes. Both the random forest and XGBoost model were best able to predict these two classes, despite them not being the most frequent classes. The algorithm occasionally ranks instances significantly worse than the benchmark. But overall the model provides better predictions on average. This can be observed in Figure 9.3 and Figure 9.4. The average ranking score per class for the 7 most historically frequent for a random test set ($N = 534$) are given below:

Benchmark	1	2	3	4	5	6	7
RCF Model	2.022	3.353	2.358	2.063	3.542	4.966	5.917

An example of the results provided by the RCF prediction model is shown in Appendix C.

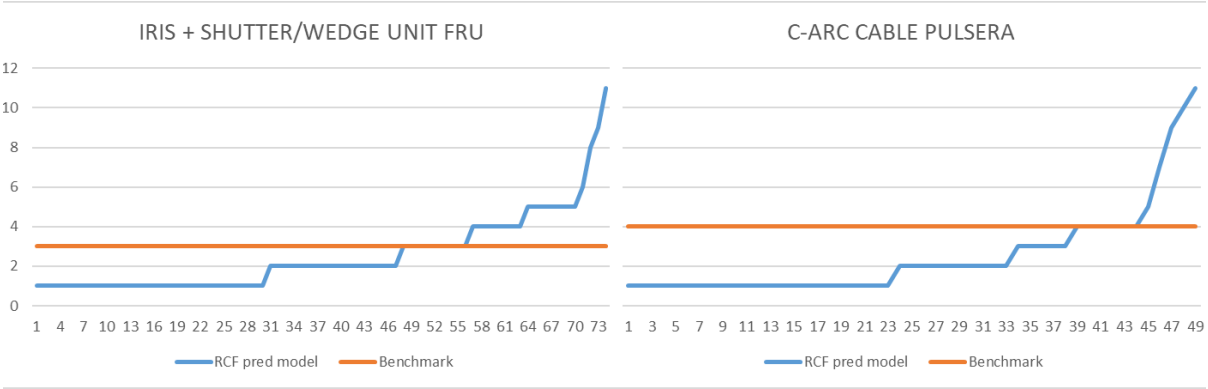


Figure 9.3 – C-arm stand: predicted rank of all 'collimator' and 'C-arc cable pulsera' instances

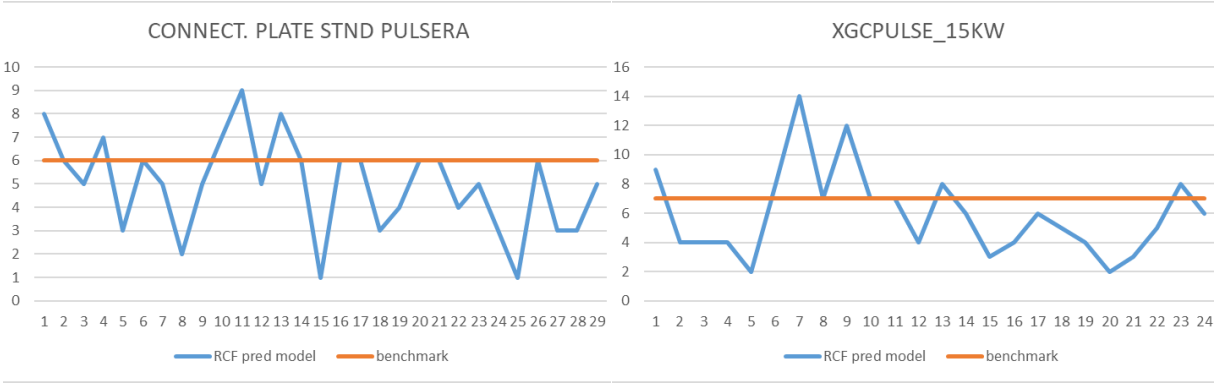


Figure 9.4 – C-arm stand: predicted rank of all 'CONNECT. PLATE STND' and 'XGCPULSE_15KW' instances

Figure 9.5 below graphs the performance of the XGBoost model in terms of the second scoring metric, which is measured as the proportion of instances for which the true RCF is ranked in the top n. The performance of the RCF prediction model is consistently better than the benchmark in terms of the second scoring metric, regardless of n.

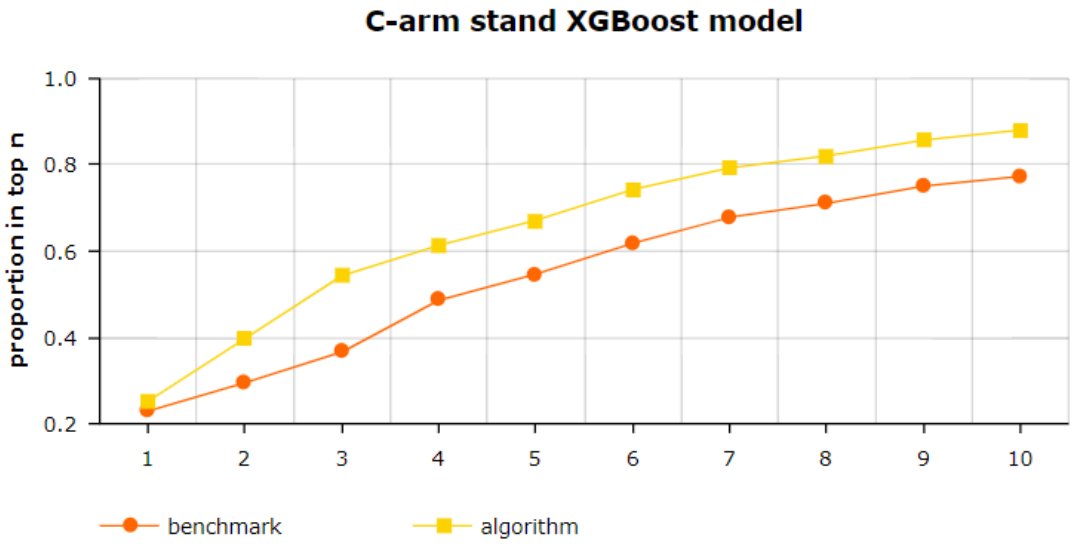


Figure 9.5 – C-arm stand: proportion of instances where the true RCF is ranked in the top n

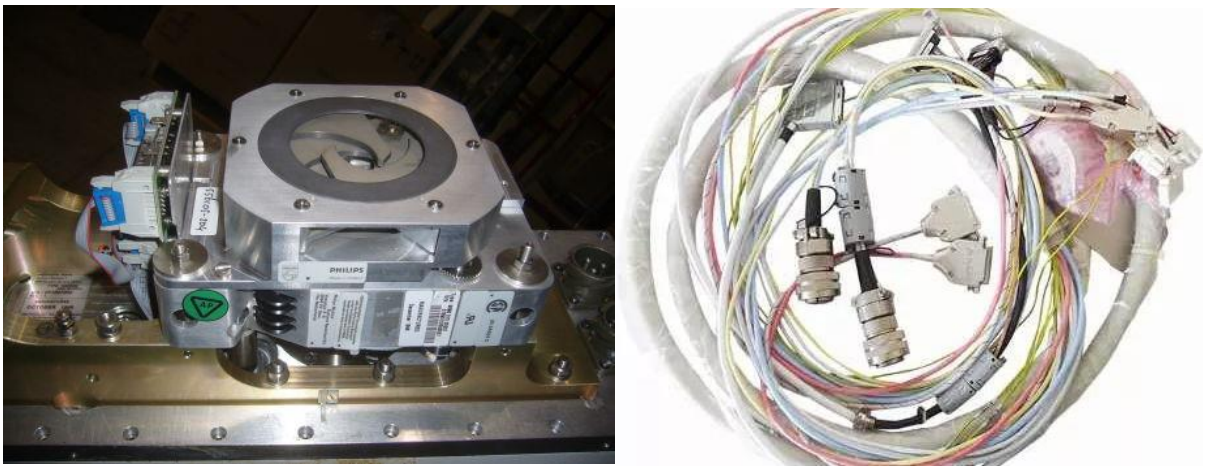


Figure 9.6 – Pictures of the Collimator (iris + shutter/wedge unit) and C-arc cable Pulsera

9.3.3. *SystemCode*=728231, CT modality

During the modelling phase described in Section 8, we zoomed in to a handful of subsystems of the iXR modality in order to allow for more extensive analysis. Recall that the objective was to develop a method applicable to all systems and parts of all modalities. Now that the model has been largely defined, it is also important to investigate its performance for different modalities.

Thus we also investigate model performance for CT modality. More specifically, the next experiment targets the 'Brilliance CT 64 CHANNEL' system type with *SystemCode*=728231. No usage prediction model was trained for this modality. Only maintenance data was used. The components considered were not restricted to a single subsystem. Instead the 50 most frequently replaced components were investigated. The 50 *ReplacementChains* contain 142 unique Part12NSs, which are found in 65 system types. Encompassing 19672 individual systems. A part list similar to Table 9.6 can be found in Appendix B. The 52437 calls involving the 50 components are reduced to 41433 calls after cleaning steps, such as the removal of duplicates and systems with limited data availability.

Approximating the age of a part

As explained in Section 8.2.2, the age of parts were approximated using the following method:

$$Age = \begin{cases} time\ in\ days\ since\ last\ replacement, & casesRecorded > 0 \\ time\ in\ days\ since\ start\ window\ of\ observation, & casesRecorded = 0 \end{cases}$$

The start of the window of observation is calculated as the earliest call recorded in data. Other alternatives, and the reason they were not used, are explained in Appendix A. When calls involving the part have been recorded, one can approximate the age of a part as either the time since the part has last been replaced (involved in a call where $Quantity > 0$), or since the last time a call was recorded involving that part (call where $Quantity$ is unrestricted). In the latter case, the assumption that repairs are perfect is implied. No significant difference in performance was observed between the methods using 5-fold cross validation.

Optimal model configurations, CT (high data availability) vs iXR (moderate data availability)

Several model configurations were tested for the CT system. The results on test datasets ($N = 4433$) are shown below in Table 9.8. Multiple conclusions can be drawn from the first four rows. The XGBoost algorithm appears to be the more accurate option for the meta-learner. Whether the survival curves are obtained using the KME or CoxPH models, the XGBoost meta-learner outperforms the RF meta-learner, which somewhat contrasts the findings of Sections 8.3.2 and 9.3.1. No meta-learner significantly outperformed the other for the UI modules subsystem of the Pulsera BV systems.

Table 9.8 – Results 'Brilliance CT 64 CHANNEL'

Survival method	Meta-learner	algorithm rankScore	benchmark rankScore	algorithm % not in top 5	benchmark % not in top 5
CoxPH	RF (mtry=16)	15.49326	19.09723	71.12285	79.65654
CoxPH	XGBoost	14.09194	19.09723	65.65390	79.65654
KME	RF (mtry=16)	16.35641	19.09723	74.55746	79.65654
KME	XGBoost	14.88560	19.09723	69.51123	79.65654
CoxPH	Homogeneous One-vs-all XGB	14.35890	19.09723	67.62915	79.65654

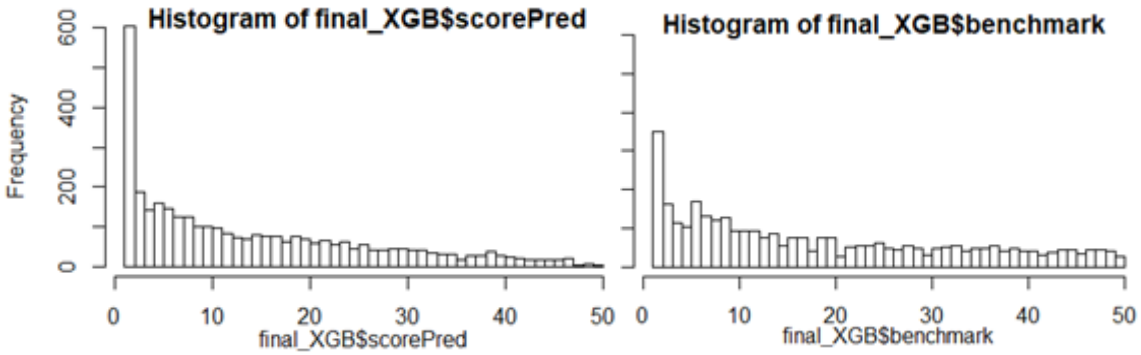


Figure 9.7 – Histogram scores CoxPH+XGBoost algorithm and benchmark on test dataset

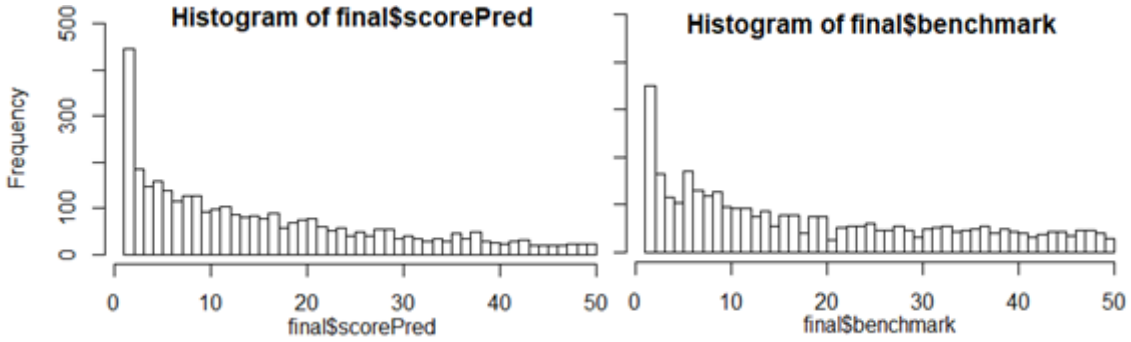


Figure 9.8 – Histogram scores CoxPH+RF algorithm and benchmark on test dataset

Histograms of the ranking scores achieved by the algorithm and the benchmark are shown in Figures 9.7 and 9.8 above for a random testset ($N = 4433$). A greater positive skew was observed for the algorithm with an XGBoost meta-learner. Most notable is the difference in

accuracy (proportion of cases where the true RCF was ranked first), ~600 for XGBoost vs ~450 for RF. An example of the output is shown Appendix D.

The 'Brilliance CT 64 CHANNEL' is one of the most popular, and thus data-rich system types in terms of maintenance data. The data availability is much higher for the CT experiment than the iXR experiment. The complexity of the problem at hand is also greater (50 vs 4 classes). It seems XGBoost outperforms RF as data availability and problem complexity increases. Besides better performance, the training time of the XGBoost algorithm is much shorter for larger datasets. Multiple papers have proven the XGBoost algorithm to be more scalable (T. Chen & Guestrin, 2016).

Furthermore, model performance was greater when CoxPH regression was applied instead of the KME, regardless of the meta-learner algorithm. Once again, the difference between the two methods is much greater than observed during the iXR experiments. Despite the lack of an *usageIntensity* covariate, CoxPH proved the more accurate method. Thus, we conclude the following. As data availability increases (both in terms of number of covariates and sample size), greater performance will be achieved using the CoxPH regression instead of the KME. As data availability and problem complexity increases, XGBoost will gradually outperform the RF algorithm to an increasing degree.

Once again, an One-Vs-All multistage approach is explored. Due to the large number of classes in the problem at hand, stages are kept homogeneous, meaning the same algorithm is used at each stage. In this case, boosted logistic regression trees of varying depths were trained. The optimal hyperparameter settings were selected using grid search. As each individual model returns comparable output probabilities (predicted value of y), ranking can be applied by directly ordering on the predicted probabilities.

The resulting ranking score is comparable to the multiclass XGBoost model, but a much quicker training time was observed. The training time of individual Logit trees is very short, and the training time of the entire multistage model increases linearly as the number of classes increases. For complex (sub) systems or fault modes (numberOfClasses>50), this meta-learner is the most scalable model, with reasonably high performance.

10. Influence of data availability on performance

In this section, the performance of the RCF prediction model is evaluated on its performance in different circumstances. The answer to the final research question will mainly be derived from this section. The section is concluded with a comparison of the performance of the model on artificial data and a conclusion regarding its sensitivity will be drawn. A multiclass XGBoost model used during the entire experiment.

10.1. Average system age

For the next experiment, the CT dataset was split up into 4 comparably sized datasets on the SystemAge variable. The average SystemAge of each subset becomes increasingly large. Most parts in the system are very reliable, failing rarely. Thus the average age of parts in the system is higher when taking a random maintenance call from the last subset. When a call relates to a very old system, some of its parts will be very old as well. More parts will be in the unstable right tail of their individual survival curves. Table 10.1 below shows the average scores obtained after 5-fold cross validation repeated two times

Table 10.1 – Results: average system age experiment

SystemAge	**0-1095**	**365-1460**	**1460-2500**	**2500-5000**
N	13177	14146	13227	10128
Benchmark rankScore	18.729	17.967	19.141	18.802
Benchmark % not in Top3	0.611	0.859	0.866	0.869
Algorithm rankScore	14.361	14.506	15.146	14.367
Algorithm % not in Top3	0.392	0.778	0.783	0.789
Shannon entropy: H(y)	3.676249	3.718331	3.777734	3.711120

For the first 3 datasets, the ranking score of the algorithm gradually worsens as the average systemAge in the dataset increases, as expected. However, the rankScore improves for the last dataset, which contains only the oldest systems. This result contradicts the hypothesis.

Possible explanations for these findings could be a less chaotic distribution in the 4th dataset, compared to the first 3. In order to investigate if this is the case, Shannon's entropy from information theory is used once again. This time to quantify the amount of uncertainty in the probability distribution of the y variable of each dataset. It is arguably harder to obtain a good ranking score when the entropy is higher. For example, imagine a classification problem with the following two labelled datasets ($N = 10$). It is harder to correctly guess the label for an instance randomly sampled from the first dataset than the second dataset:

$$a, a, a, a, a, b, b, b, b, b\ (entropy = 1) \qquad a, a, a, a, a, a, a, a, a, b\ (entropy = 0.47)$$

The final row in Table 10.1 contains the entropy of the y variable. One observes a slight correlation between the algorithm rankScore and the entropy of the y variable. Paired with the slight differences in scores observed, no conclusions can be drawn with confidence regarding the effect of systemAge on model performance. No threshold for model usefulness could be determined, nor a quantification of the loss of performance due to the increase of systemAge.

10.2. Rate of censoring

In order to investigate the relation between the censoring rate and model performance, a percentage up to 80 of uncensored events in the data used to construct the survival curves were artificially censored. The survival curves are constructed using either CoxPH regression

or the KME, with the hyperparameter settings as specified in Sections 7.1.1 (*repairIntensity* is approximated over the entire time the system was observed in the data) and 8.2.4 (DeltaRange=125). The meta-learner used is a 50-class XGBoost classifier. Table 10.2 below summarizes the results of the experiment.

Table 10.2 – Results: rate of censoring 20%-80% experiment

Additional censoring	Survival method	algorithm rankScore	benchmark rankScore	algorithm % not in top 5	benchmark % not in top 5
20 %	CoxPH	14.26979	18.72569	0.6753891	0.7917889
20 %	KME	15.02211	18.72569	0.6918565	0.7917889
40 %	CoxPH	14.35439	18.72569	0.6749380	0.7917889
40 %	KME	15.03136	18.72569	0.6959170	0.7917889
60 %	CoxPH	14.37040	18.72569	0.6823821	0.7917889
60 %	KME	15.02278	18.72569	0.7015565	0.7917889
80 %	CoxPH	14.44958	18.72569	0.6972705	0.7917889
80 %	KME	15.07918	18.72569	0.6923077	0.7917889

A barely noticeable effect was observed on the performance of the algorithm for the KME. A more noticeable effect was observed when CoxPH regression was applied. This results was somewhat to be expected as the curves are primarily calculated using censored events from the start. In addition, each curve was subject to the same proportion of additional censoring.

Note that the 'time' variable was left unchanged. A different approach was thus also investigated, where a proportion of the events are sampled and removed entirely. Once again as shown in Table 10.3, the performance of the algorithm using both survival methods deteriorated only slightly when removal rate ≤ 80, albeit faster than when only the 'status' variable was changed.

Table 10.3 – Results: rate of removal 20%-80% experiment

removal rate	Survival method	algorithm rankScore	benchmark rankScore	algorithm % not in top 5	benchmark % not in top 5
20 %	CoxPH	14.25874	18.72569	0.6756147	0.7917889
20 %	KME	15.01624	18.72569	0.6956914	0.7917889
40 %	CoxPH	14.31311	18.72569	0.6765170	0.7917889
40 %	KME	15.03474	18.72569	0.6954658	0.7917889
60 %	CoxPH	14.43379	18.72569	0.6778705	0.7917889
60 %	KME	15.03293	18.72569	0.6959170	0.7917889
80 %	CoxPH	14.49831	18.72569	0.6799007	0.7917889
80 %	KME	15.09948	18.72569	0.6911798	0.7917889

More interesting results were observed as the removal rate approached 80%. At this point, the 50 individual survival curves are calculated using 493 observed individuals on average, of which 156 are uncensored. 6 out of the 50 parts were sampled and plotted in Appendix F. The survival curves drawn with 80% removal still closely resembled those drawn without artificial removal. Resemblance started to wane after 95% removal.

Table 10.4 contains several measurements obtained during a final experiment. During this experiment, the removal rate was gradually increased up to 95% as indicated in the first column. The KME was tested for all rates of removal, while the CoxPH was investigated only when the removal rate was 95% and 99%, due to time constraints. The scores obtained using the KME are shown in the last two columns, at the top of each row. The scores obtained using CoxPH are shown at the bottom of each row, between parentheses ().

Table 10.4 – Results: rate of removal 95%-99% experiment

removal rate	**Uncensored observations**			**Total observations**			**algorithm rankScore**	**algorithm % not in top 5**
80 %	Min 31	Mean 156	Max 598	Min 164	Mean 493	Max 1675	15.09948 (14.49831)	0.6911798 (0.6799007)
95 %	Min 8	Mean 37	Max 150	Min 41	Mean 123	Max 419	15.11730 (14.55425)	0.6954658 (0.6866682)
96 %	Min 6	Mean 30	Max 121	Min 33	Mean 99	Max 335	15.13377	0.6916309
97 %	Min 4	Mean 24	Max 95	Min 25	Mean 74	Max 251	15.23889	0.6972705
98 %	Min 3	Mean 17	Max 64	Min 16	Mean 49	Max 168	15.26619	0.7017821
99 %	Min 1	Mean 9	Max 34	Min 8	Mean 25	Max 84	15.32123	0.6965937

Several descriptive statistics regarding the number of observations used to fit the survival curves are given in the 2nd to 7th columns. Let N be the average number of observations used to fit a survival curve. One can see that $N = 25$ during the final experiment. It is stated in literature that CoxPH generally requires more data/observations to reach accurate results than the KME (Kalbfleisch & Prentice, 2011). This led to the following hypothesis: ''when defining the performance of the algorithm as a function of N, there exists an N where the algorithm performance is higher when the KME is used instead of CoxPH regression. Unfortunately, this hypothesis could not be tested empirically due to time constraints.

11. Summary

The results of this research are provided in this chapter in the form of answers to the research questions. These answers are directly derived from sections 4 to 10. The answer to the main research question is derived from all answers combined.

11.1. Research question 1

What data relevant to Medical Imaging Systems maintenance is currently gathered by Philips?

Data supporting the maintenance of MISs are currently gathered and combined in the ISDA database in a Vertica database solution. Two main types of data were identified, namely maintenance and usage data. However, only maintenance data is universally available for all system types and parts, regardless of software and hardware versions. As explained in Section 5, several variables of interest are identified. Due to the objectives set for this project, focus is placed on universally available data. Section 6 explains the steps needed to prepare the data for survival analysis. The storage of relevant data in multiple tables elicits the need for multiple aggregation steps. Feature selection was performed using the Log-rank test, together with the MI metric from information theory.

Despite usage data being limitedly available for most systems, the presence of a few system types fir which usage data is faithfully recorded allows for stacking ensemble methods to be used to approximate the usage of system without available usage data.

11.2. Research question 2

Can survival analysis be used to distinguish between heavily and lightly used individual systems of any system, using only part replacement data?

As described in Section 7, structured usage data detailing exams performed is recorded for approximately 44 system types of the IXR module. Data was most faithfully recorded and numerous for the [redacted] systems. The *SystemCode* belonging to these systems are [redacted]. Cleaning and transforming the data reduced the number of systems in the dataset from around 1500 to around 1200 systems.

Several statistical tests and methods from information theory affirmed a weak correlation between the usage and repair intensities of individual systems. Greater information gain was obtained from corrective replacement data when predicting the total number of hours in operation, when compared to predicting the number of exams overall. Corrective replacements (calls involving a *Part12NC* & *Quantity*='1') were found to be a better predictor for usage than general repairs (calls with any *Quantity* variable).

REDA and multiple machine learning techniques were evaluated on their ability to predict the *usageIntensity* of a system using data obtained from the maintenance database. Both a classification and regression model were trained, each providing a covariate for different

survival analysis methods. Systems were classified on whether their *usageIntensity* was higher or lower/equal than the median. The best performing models were a random forest classifier and random forest regression model.

In Section 8, a substantial increase in model accuracy was observed during the evaluation of the RCF prediction model when provided with the usage prediction model. This was the case both when the KME was applied, as well as when CoxPH regression was applied. A smaller but still noticeable increase in accuracy was also observed when only the *repairIntensity* covariate, obtained using REDA, was included.

Thus, it is reasonable to conclude that survival analysis can be used to distinguish heavily used system from lightly used systems, using only replacement data. However, survival analysis is outclassed by random forest and other general-purpose machine learning techniques for this purpose. Features that should be included in order of descending importance, measured by both the importance score (Gini-impurity) from random forest and information theory:

- *numberOfCorrectiveReplacements (Quantity of call =1)*
- *SystemCode*
- *MarketGroup*

Furthermore, interest was expressed in using survival analysis for determining which individual parts are subject to wear, and to what extent. This knowledge can naturally be derived from assessing CoxPH regression models for individual parts that include the *usageIntensity* covariate. After assessing the fit of the CoxPH models using a multitude of metrics and statistics, the β relating to the *usageIntensity* covariate can be interpreted as the isolated effect of *usageIntensity* on the hazard of a part.

However, sufficient usage data is required in order to obtain is *usageIntensity* covariate. Section 7.1 showed that this data is limitedly available for most system types. When using only maintenance/part replacement data, the usage prediction models described in Section 7.4 can be used to obtain an approximation of the *usageIntensity* variable. The application of this model greatly reduces the certainty of statements made based on the fitted CoxPH model, as the *usageIntensity* prediction model introduces a significant level of bias and uncertainty.

11.3. Research question 3

Can survival analysis methods be used to support root-cause analysis at Philips?

The following three research gaps were identified in Section 4, shaping the main objective of this paper: (1) Use of survival analysis for RCA, (2) Multi-component system maintenance models dealing with censoring, (3) Data-driven RCA. The conceptualization and evaluation of a root-cause prediction model based on survival analysis for the support of RCA, was set as the main objective of this paper in order to expand public knowledge on the identified domains.

The conceptual framework for the model was formalized in Section 8, followed by the annals of modelling decisions made and the intermediate results on which they were based. A series of experiments were conducted, leading to the identification of multiple model settings, none of which dominated all others in every aspect and metric.

Ranking parts using only a single predictor derived from survival curves outperformed the benchmark only in a few cases, and only when the age of systems were constrained. It was noted that different predictors performed better in certain instances. It was concluded that survival analysis alone was not enough to accurately approximate the failure behaviour using the available data. Thus the final modelling decision explored was the appendation of a meta-learner model that makes a final prediction using the survival curve based predictors as features, in order to make the algorithm more robust to scarcity and noise in the available data. Two archetypes of meta-learner models were presented; multiclass classifiers and cascades of (heterogeneous) binary classifiers.

Section 9 presents the evaluation of the best models selected in Section 8. Experiments were limited to two system types from the iXR and CT modalities for which the most data was available in the *catmasterlist* table. The survival analysis method used for the final evaluation was the CoxPH model. Predictors were obtained as follows:

- The slope of the survival curve at a certain point in time t was approximated by averaging the decrease in survival rate over a period of 250 days around t.
- *RepairIntensities* of systems were approximated over the entire observation window.

Performance of different archetypes of meta-learners were evaluated for the 'UI modules' subsystem. Multiclass classifiers (0.539) were outperformed by a cascade of binary classifiers using a One-Vs-All approach (0.493). The 'threshold for labelling' of these binary classifiers was set to be strict. Meaning that an instance is only labelled as the 'positive' class when the model is certain. The best performance was observed when a combination of both was used (0.444). Translating these findings to practice, one would for a given case; 1) first apply binary classifiers until the case is predicted to be of the 'positive' class. 2) If after applying all trained binary classifiers the case is still predicted to be of the 'Other' class, one applies the multi-class classifier.

For the C-arm stand subsystem, the best performance was observed with an XGBoost meta-learner (4.763). The RF meta-learner (4.973) also outperformed the benchmark (5.504).

Several metrics and statistics can be used to select the best model for acting as the meta-learner model. In Section 9, the model performance in terms of the defined metrics was used, e.g. average ranking score and proportion of instances for which the true RCF was ranked highly. However, there are numerous alternatives that could be used such as the AIC, BIC, Kullback–Leibler distances and F-score. More obviously, one can use validation methods such as cross validation to determine the best model.

The resulting algorithm can be summarized as follows:

Algorithm: root-cause prediction model

Input: maintenance call requiring corrective replacement + *subsystem (optional)*

Obtain set *P* of individual parts *p* in the *subsystem* and their age *t:*

$$t \leftarrow \begin{cases} time\ in\ days\ since\ last\ replacement, & casesRecorded > 0 \\ time\ in\ days\ since\ start\ window\ of\ observation, & casesRecorded = 0 \end{cases}$$

Obtain $characteristics_{system} = (market \quad part12NC \quad repairIntensity \quad usageIntensity)$

$$repairIntensity(t) \leftarrow \begin{cases} \frac{\sum_{t-\tau \le i: t_i \le t}(Z_i)}{\tau}, & t - \tau \ge 0 \\ \frac{\sum_{i: t_i \le t}(Z_i)}{t}, & t - \tau < 0 \end{cases}$$

$$usageIntensity \leftarrow \frac{\sum_i^N \frac{Y_i}{3600}}{t_N - t_1 - \sum_i^{N-1} Gap(i)} \textbf{ or } \text{pred}(usageIntensityModel, newdata = characteristics_{system})$$

for *p* = 1 ***to*** *P* **do**

| **query all historical calls and pre-process the data**

| **fit** *CoxPH* model for part *p*

| $L(\boldsymbol{\beta}) = \frac{\prod_{i \in D(t)} \exp\ [\boldsymbol{\beta} * X_i(t)]}{\prod_{h=1}^{d(t)} \left(\sum_{i \in R(t)} \exp\ [\boldsymbol{\beta} * X_i(t)] - \frac{h-1}{d_i} \sum_{k \in D(t)} \exp\ [\boldsymbol{\beta} * X_i(t)] \right)}$

end

derive survival curve using KME or *CoxPH* model

$$\hat{S}(t)_p = \prod_{i: t_i \le t} \left(1 - \frac{f_i}{r_i}\right)$$

for *p* = 1 ***to*** *P* **do**

| **calculate** *survival curve predictors*

| $S(t)_p \leftarrow \hat{S}(t)_p$

| $\widehat{\Delta S(t)}_p \leftarrow \begin{cases} \frac{S(t-DeltaRange)_p - S(t+DeltaRange)_p}{2 * DeltaRange}, & t - DeltaRange \ge 0 \\ \frac{1 - S(t+DeltaRange)_p}{t + DeltaRange}, & t - DeltaRange < 0 \end{cases}$

| $RA_p \leftarrow \frac{S(t)_p^2}{2 * \widehat{\Delta S(t)}_{\mathrm{p}}}$

end

select $metaLearner$: $AIC, BIC, CrossValidation, \quad \arg\min_{metaLearner \in allModels} (score_{avg})$

for *p* = 1 ***to*** *P* **do**

| $\mathbf{Prob\{p = RCF\}} \leftarrow \text{pred}(metaLearner, newdata = [survival\ curve\ predictors]_{\mathrm{p}})$

end

Output: List of parts $p \in P$ ranked on their probability of being the RCF

It is difficult to say to what extend survival analysis can be used to improve the RCA process in the maintenance of MISs, due to the lack of a widely accepted performance metric for the problem presented. For the BV Pulsera systems, the RCF prediction modal based on survival analysis does seem to outperform the simplest benchmark for certain subsystems. A greater degree of improvement is observed for the more data-rich Brilliance CT 64 CHANNEL system.

11.4. Research question 4

To what extent are the results of the analysis answering the third research question dependent on the system and part type in question, in regards to data availability?

As data availability increases (both in terms of covariates and sample size), greater performance will be achieved using the CoxPH regression instead of the KME. As data availability and problem complexity increases, XGBoost will gradually outperform the RF algorithm as the meta-learner, to an increasing degree. Besides better performance, the training time of the XGBoost algorithm is much shorter for larger datasets. With regards to scalability for complex problems (>50 classes), boosted logit trees using a One-Vs-All approach is the best option for the meta-learner, with little loss in predictive power.

Let N be the average number of observations used to fit a survival curve. Experimentation with artificial data reduction was conducted in pursuit of a 'tipping point' where the KME outperformed CoxPH regression. Additional censoring was found to have little effect on algorithm performance. The tipping point was not found for the CT system fault mode investigated, until additional removal reached 99% ($N \approx 25$) of all observed events. Algorithm performance deterioration became more significant as N decreased. This effect was greater for CoxPH regression than the KME. Plateauing of algorithm performance as a function of N was observed as N approached 400, when using CoxPH+XGBoost.

Model scalability

The practical value of a model is not solely determined by its accuracy in terms of variance and bias. There is also model interpretability, maintainability and training time. The effect of the latter on model value is amplified by the dynamic nature of the problem at hand. Data is continuously added to the database. In order to take advantage of this data, models should occasionally be refitted. Because of this, training time plays a big role in the maintainability, especially when more complex models are used.

Figure 11.1 illustrates the difference in time required to obtain the training datasets for the meta-learner when using either the KME or CoxPH method. After fitting the survival curves beforehand, obtaining the CT dataset ($N \approx 42{,}000$) using the KME took approximately 3 hours. After fitting the individual CoxPH models beforehand, obtaining the CT dataset ($N \approx 42{,}000$) using CoxPH regression took approximately 25 hours. Time querying Vertica should have a negligible effect on the total runtime, as only two small queries per call were executed. The difference in runtime can be When using the KME, the final survival curves can be stored directly. When using CoxPH

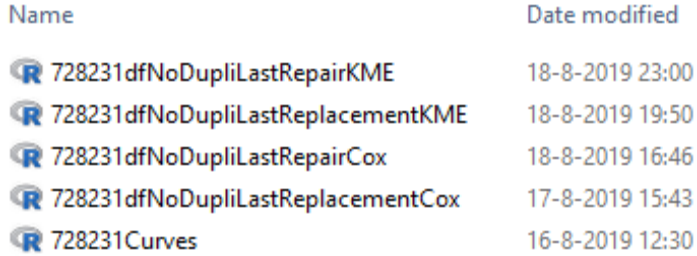


Figure 11.1 – Runtimes obtaining meta-learner training datasets

regression however, one should store the fitted regression models quantifying the hazard rates. Only after $characteristics_{system}$ (market part12NC repair usage) is obtained, are the survival curves drawn. Drawing these curves and storing them is not really practically feasible due to the large number of possible 'profile' permutations.

11.5. Problem statement

How is the maintenance process of medical imaging systems currently supported by data gathered by Philips and can this process (involving any part) be supported further using survival analysis?

The maintenance process of MISs is currently supported by multiple data-driven prediction models. These model achieve great accuracy in predicting imminent failure and assessing system health. The majority of these models are focussed on improving PM, are specialized for one part and require data that is not always available for the system in question.

CM still forms the significant majority of maintenance calls. Currently, the RCA domain is dominated by knowledge-driven methods such as the 5 why's and the fishbone diagram. With MISs generally consisting of thousands of precisely interacting parts, it is not surprising that the number of failure modes is mind boggling. For a significant proportion of corrective maintenance calls, imminent failure of the part in question had already been flagged by prediction models supporting PM. Diagnosing the root-cause is a simple affaire in these cases. For some cases however, the root-cause is not immediately clear, and requires a service engineer to diagnose the system on-site.

For these cases, service engineers strive to fix the system in one visit. Assuming the system failure is caused by the failure of a single component, survival analysis-based RCA can be used to narrow down the list of possible failure modes. Preliminary RCA could expedite the diagnosis process, which results in a lower costs incurred due to downtime, decrease in necessary inventory levels and higher probability of first-time right visits.

A RCF prediction model using universal data was evaluated and found to outperform a simple baseline method to a moderate degree. The model performance does not compare to more specialized models that use more specific data and health checks. But RCA based on survival analysis could serve as a general go-to tool for preliminary analysis, only drawing a conclusion when it is confident about its prediction. Should this not be the case, further analysis using more specific models should follow.

The optimal model configuration differs per problem such as the (sub)system in question. Different (survival or machine learning) methods should be selected, depending on the data availability, class distribution etc. For example, the model should be constructed of different elementary models, e.g. usageIntensity prediction models/meta-learners, when the problem at hand regards an iXR system, than when a CT system is analysed.

12. Discussion

This research project encompasses all phases of the CRISP-DM methodology, minus the deployment phase. The main objective of the thesis is to serve as a proof of concept of the RCF prediction model and analysing the possibilities that survival analysis bring, with regard to multi-component system maintenance without assuming complete information. Therefore, the intended audience consists of scholars working on extending knowledge on these domains. As for the contributions of the findings towards data driven root-cause analysis, most details of the proposed model are not generalizable outside Philips.

A larger part of the findings are relevant to the researchers at Philips who seek to optimize the maintenance of MISs, possibly through the implementation of a data-driven RCF prediction model based on survival analysis. Ideally, this paper will form the starting point of that process and allow Philips to make even further use of their data.

Things to consider when considering the implementation of a RCF prediction model for a specific subsystem:

- The evaluation metrics used are a point of great uncertainty. These were arbitrarily chosen and more specific alternatives derived from expert opinion should be contrived.
- Additional non-universally available variables should be considered, such as table movements when the subsystem in question is the 'table' subsystem.

12.1. Limitations

This research also has its fair share of limitations worth mentioning. The objective of this paper was to answer the question: 'What can be done with survival analysis as a general method?' Focussing on a specific part, system type or subsystem was avoided where possible. When focussing on a certain subsystem proved unavoidable, methods and data used were limited to those generalizable and universally available for every subsystems. The project was also primarily exploratory as it did not build further upon previous projects conducted at Philips. Because of this, little focus was placed on practical applicability and contact with the engineers in Best was limited.

Parametric methods

There exist numerous parametric alternatives to the CoxPH for approximating the hazard rate of a machine component that have been completely disregarded during this project, due to the high degree of censoring observed. Models such as those based on the Weibull or exponential distribution have proved their value time after time and might warrant research in the future.

Extensively investigated modalities/system types

From the start, the goal of a general tool was stated. A method supporting the maintenance of MISs of any modality or system type. However, as experiments and analysis deepened, only

2 or 3 subsystems within a single modality were investigated extensively. Usage approximation was only tested for the IXR modality, and training meta-learner models was limited to two subsystems from the Pulsera BV systems and the Brilliance CT 64 CHANNEL systems. Aside from the *usageIntensity* variable, all data used were confirmed to be available for the MR and CT modalities as well. Then again, these modalities were not investigated extensively.

12.2. Future research

An obvious direction of future research is inducing a project encompassing the entire CRISP-DM methodology, including the final step. Should Philips deem the results interesting enough to warrant further research, they could start a new project that encompasses the entire methodology anew, while focussing on a specific subsystem or set of parts.

Batching maintenance activities in multi-component systems

The maintenance policy described by Van Horenbeek & Pintelon (2013) is of great interest to the researcher as it provides a framework for applying the RCF prediction model, and evaluating the possibility of saving costs. The basic premise of the policy is centred on the existence of economic dependencies and the possibility to save cost by grouping maintenance of multiple components. Potential is seen in the RCF prediction model as a heuristic tool enabling opportunistic maintenance.

Considering a MISs consisting of multiple non-identical components, it is assumed a failure of any individual component will bring the system to a halt. Failure of the system is instantaneously noticed without inspection.

Van Horenbeek & Pintelon (2013) tersely describe a dependence parameter α_d, which should reflect the economic and structural dependence of multiple components. Unfortunately, obtaining such a parameter is no trivial task and requires significant assumptions. Given the complexity of MISs, the presence of this step heavily reduces the feasibility of applying the proposed policy. This step can be supplanted by the RCF prediction model.

When the system grinds to a halt due to failure of an unknown component, data is gathered and the RCF prediction model is applied, returning a list of predictions/remaining lifetime approximations. This allows the designation of a subset of components, likely to be the root-cause of the system failure. Spares of these components are brought along by the remote service engineer. This simplifies the problem of determining the dependence parameter. Individual components whose prospective remaining life exceed a dynamic threshold are replaced preventively. The incorporation of the RCF prediction model in the policy greatly reduces the complexity and computational requirements when the number of non-identical components in the system exceeds tens.

Unfortunately, evaluating such a policy supported by the RCF prediction model, even with simulated data, was unfeasible within the time constraints of this project.

Focussing/expanding the scope of the problem

Focussing on a specific system type allows one to be much more inquisitive about what data to include. When researching the applicability of a RCF prediction model for a specific system, one should consider the actual costs associated with bringing certain parts to the client, as well as exclude easily diagnosable components from the analysis. As mentioned before, future researchers should spend time researching and categorizing interesting parts based on their volume, diagnosability and cost. Easily diagnosable and low cost parts such Ethernet cables, keyboards, mice and dongles should be excluded from the analysis. Future projects might also cast of the constraints of the subsystem variable, instead observing a set of interesting parts within a system type.

Possibilities of combining the results from the RCF prediction model with existing algorithms and methods is not limited to setting up maintenance policies. The possibilities are numerous. As an example:

Service engineers strive to solve maintenance calls in one visit, e.g. they strive for 'first-time right'. The question of the service engineer regarding what components to bring to the client could be tackled as a classical knapsack problem using the RCF prediction model. The volume/weight of parts and capacity of the knapsack (service van) are straightforward in this scenario. These parameters can also be replaced by the total value of parts to be shipped. Instead of each item/component having a monetary value, it is assigned a probability of being the RCF by the RCF prediction model. Naturally, the objective would be to maximize the probability of the true RCF being chosen.

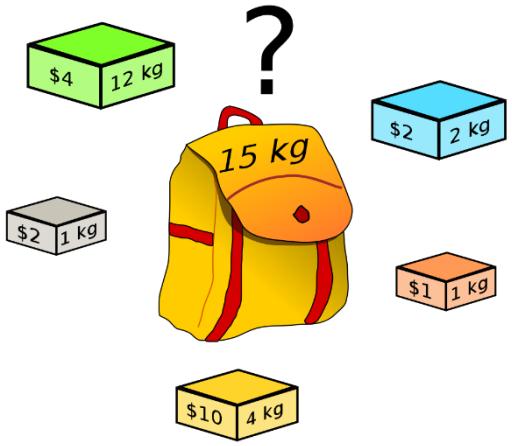


Figure 12.1 – Illustration of knapsack problem

Furthermore, the method of using predictors derived from survival analysis warrants further investigation, possibly with simulated data in order to truly understand the relationships between data availability and the accuracy of the method described. Too many assumptions are required when using the actual ISDA data to draw reliable conclusions about the intricacies of the method.

Practical application was purposely left out of scope during this project. In future projects, closer cooperation with the engineers in Best is imperative. Next to being an obvious stakeholder, their expertise form a key source of information regarding many issues that were glossed over with assumptions in this project. While volume and costs of parts can be obtained from data, the engineers would provide the best bet for obtaining diagnosability. Combination possibilities with existing models are necessary as well if practical value is the goal. Their insights are vital to designing the right metrics for model evaluation.

References

Aalen, O. (1978). Nonparametric inference for a family of counting processes. *The Annals of Statistics*, 701–726.

Aljawadi, B. A. I., Bakar, M. R. A., & Ibrahim, N. A. (2012). Nonparametric versus parametric estimation of the cure fraction using interval censored data. *Communications in Statistics-Theory and Methods*, *41*(23), 4251–4275.

Andersen, P. K., & Gill, R. D. (1982). Cox's regression model for counting processes: a large sample study. *The Annals of Statistics*, 1100–1120.

Arthur, V., & Peterson, J. (1977). Expressing the Kaplan-Meier Estimator as a Function of Empirical Subsurvival Functions. *Journal of the American Statistical Association*, *72*(360a), 854–858. https://doi.org/10.1080/01621459.1977.10479970

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123–140.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32.

Breiman, L. (2004). Consistency for a simple model of random forests.

Breslow, N., & Crowley, J. (1974). A large sample study of the life table and product limit estimates under random censorship. *The Annals of Statistics*, 437–453.

Brodersen, K. H., Ong, C. S., Stephan, K. E., & Buhmann, J. M. (2010). The balanced accuracy and its posterior distribution. In *2010 20th International Conference on Pattern Recognition* (pp. 3121–3124).

Bylander, T. (2002). Estimating generalization error on two-class datasets using out-of-bag estimates. *Machine Learning*, *48*(1–3), 287–297.

Chawla, N. V, Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, *16*, 321–357.

Chemweno, P., Morag, I., Sheikhalishahi, M., Pintelon, L., Muchiri, P., & Wakiru, J. (2016). Development of a novel methodology for root cause analysis and selection of maintenance strategy for a thermal power plant: A data exploration approach. *Engineering Failure Analysis TA - TT -*, *66*, 19–34. https://doi.org/10.1016/j.engfailanal.2016.04.001 LK - https://tue.on.worldcat.org/oclc/6019892808

Chemweno, P., Pintelon, L., Van Horenbeek, A., & Muchiri, P. (2015). Development of a risk assessment selection methodology for asset maintenance decision making: An analytic network process (ANP) approach. *International Journal of Production Economics*, *170*, 663–676.

Chen, C., Liaw, A., & Breiman, L. (2004). Using random forest to learn imbalanced data. *University of California, Berkeley*, *110*(1–12), 24.

Chen, J., & Chen, Z. (2008). Extended Bayesian information criteria for model selection with large model spaces. *Biometrika*, *95*(3), 759–771.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785–794).

Colosimo, E., Ferreira, F., Oliveira, M., & Sousa, C. (2002). Empirical comparisons between Kaplan-Meier and Nelson-Aalen survival function estimators. *Journal of Statistical*

Computation and Simulation, *72*(4), 299–308. https://doi.org/10.1080/00949650212847

Cover, T. M., & Thomas, J. A. (2012). *Elements of information theory*. John Wiley & Sons.

Cox, D. R. (1972). Regression models and life-tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, *34*(2), 187–202.

Cox, D. R. (1975). Partial likelihood. *Biometrika*, *62*(2), 269–276.

Ebeling, C. E. (2004). *An introduction to reliability and maintainability engineering*. Tata McGraw-Hill Education.

Efron, B., Hastie, T., Johnstone, I., Tibshirani, R., & others. (2004). Least angle regression. *The Annals of Statistics*, *32*(2), 407–499.

Fan, J., Li, R., & others. (2002). Variable selection for Cox's proportional hazards model and frailty model. *The Annals of Statistics*, *30*(1), 74–99.

Gamel, J. W., & Vogel, R. L. (1997). Comparison of parametric and non-parametric survival methods using simulated clinical data. *Statistics in Medicine*, *16*(14), 1629–1643.

Greenwell, B., Boehmke, B., Cunningham, J., & Developers, G. B. M. (2019). gbm: Generalized Boosted Regression Models. Retrieved from https://cran.r-project.org/package=gbm

Greenwood, M., & others. (1926). A Report on the Natural Duration of Cancer. *A Report on the Natural Duration of Cancer.*, (33).

Hothorn, T., Hornik, K., & Zeileis, A. (2006). Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics*, *15*(3), 651–674.

Johnson, L. L., & Shih, J. H. (2012). Chapter 23 - An Introduction to Survival Analysis. In J. I. Gallin & F. P. Ognibene (Eds.), *Principles and Practice of Clinical Research (Third Edition)* (Third Edit, pp. 285–293). Boston: Academic Press. https://doi.org/https://doi.org/10.1016/B978-0-12-382167-6.00023-0

Jr., A. V. P. (1977). Expressing the Kaplan-Meier Estimator as a Function of Empirical Subsurvival Functions. *Journal of the American Statistical Association*, *72*(360a), 854–858. https://doi.org/10.1080/01621459.1977.10479970

Kalbfleisch, J. D., & Prentice, R. L. (2011). *The statistical analysis of failure time data* (Vol. 360). John Wiley & Sons.

Kaplan, E. L., & Meier, P. (1958). Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association*, *53*(282), 457–481.

Liaw, A., & Wiener, M. (2002). Classification and Regression by randomForest. *R News*, *2*(3), 18–22. Retrieved from https://cran.r-project.org/doc/Rnews/

Link, C. L. (1984). Confidence intervals for the survival function using Cox's proportional-hazard model with covariates. *Biometrics*, 601–609.

Luo, S., Xu, J., & Chen, Z. (2015). Extended Bayesian information criterion in the Cox model with a high-dimensional feature space. *Annals of the Institute of Statistical Mathematics*, *67*(2), 287–311.

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., & Leisch, F. (2019). e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien. Retrieved from https://cran.r-project.org/package=e1071

Mitchell, M. W. (2011). Bias of the Random Forest out-of-bag (OOB) error for certain input

parameters. *Open Journal of Statistics*, *1*(03), 205.

Nelson, W. (1972). Theory and Applications of Hazard Plotting for Censored Failure Data. *Technometrics*, *14*(4), 945–966. https://doi.org/10.1080/00401706.1972.10488991

Nelson, W. B. (2003). *Recurrent events data analysis for product repairs, disease recurrences, and other applications* (Vol. 10). SIAM.

Nicolai, R. P., & Dekker, R. (2008). Optimal Maintenance of Multi-component Systems: A Review. In *Complex System Maintenance Handbook* (pp. 263–286). London: Springer London. https://doi.org/10.1007/978-1-84800-011-7_11

Peng, Y., & Carriere, K. C. (2002). An empirical comparison of parametric and semiparametric cure models. *Biometrical Journal: Journal of Mathematical Methods in Biosciences*, *44*(8), 1002–1014.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *ArXiv Preprint ArXiv:1609.04747*.

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

Shen, P. (2002). A NOTE ON THE HOMOGENETIC ESTIMATE FOR THE VARIANCE OF THE KAPLAN–MEIER ESTIMATE. *Communications in Statistics - Theory and Methods*, *31*(9), 1595–1603. https://doi.org/10.1081/STA-120013014

Terry M. Therneau, & Patricia M. Grambsch. (2000). *Modeling Survival Data: Extending the {C}ox Model*. New York: Springer.

Therneau, T. M. (2015). A Package for Survival Analysis in S. Retrieved from https://cran.r-project.org/package=survival

Tibshirani, R. (1997). The lasso method for variable selection in the Cox model. *Statistics in Medicine*, *16*(4), 385–395.

Urbanek, S. (2018a). rJava: Low-Level R to Java Interface. Retrieved from https://cran.r-project.org/package=rJava

Urbanek, S. (2018b). RJDBC: Provides Access to Databases Through the JDBC Interface. Retrieved from https://cran.r-project.org/package=RJDBC

van der Ploeg, T., Austin, P. C., & Steyerberg, E. W. (2014). Modern modelling techniques are data hungry: a simulation study for predicting dichotomous endpoints. *BMC Medical Research Methodology*, *14*(1), 137. https://doi.org/10.1186/1471-2288-14-137

Van Horenbeek, A., & Pintelon, L. (2013). A dynamic predictive maintenance policy for complex multi-component systems. *Reliability Engineering & System Safety*, *120*, 39–50.

Volinsky, C. T., & Raftery, A. E. (2000). Bayesian information criterion for censored survival models. *Biometrics*, *56*(1), 256–262.

Wang, W., Fu, H., & Yan, J. (2017). {reda}: {R}ecurrent Event Data Analysis. Retrieved from https://cran.r-project.org/package=reda

Wu, Y. (2012). Elastic net for Cox's proportional hazards model with a solution path algorithm. *Statistica Sinica*, *22*, 27.

Zawawy, H., Kontogiannis, K., Mylopoulos, J., & Mankovskii, S. (2012). Requirements-driven root cause analysis using markov logic networks. In *International Conference on Advanced Information Systems Engineering* (pp. 350–365).

Appendix A: [redacted]

Appendix B: [redacted]

Appendix C: [redacted]

[redacted]

Appendix D: [redacted]

[redacted]

Appendix E: [redacted]

[redacted]

Appendix F: [redacted]