Incorporating performance indicators in the decision-making process for predictive maintenance
With an application to iXR systems at Philips

Johannesson, E.G.

Award date:
2020

Master Thesis

Incorporating performance indicators in the decision-making process for predictive maintenance

With an application to iXR systems at Philips

Emil Johannesson, B.Sc.

March 22, 2020

in partial fulfilment of the requirements for the degree of
Master of Science
in Industrial and Applied Mathematics

University supervisors
Dr. Stella Kapodistria
Collin Drent, M.Sc.

Company supervisors
Maikel Boumans, B.Sc
Dr. Antonio Perrone

Executive Summary

Background Companies that produce high-valued machines are often responsible for the maintenance of the machines. The traditional way to maintain the machines is to use a corrective maintenance policy, which results in long downtimes and large costs. Policies that fall under preventive maintenance try to reduce the costs and downtime of machines. Current implementations of predictive policies are based on machine learning models that predict imminent failures. These models do not take the operational costs, i.e. the costs made following a certain policy, into account, but only focus on minimizing the number of false alerts. The aim of this project is to derive and solve an optimization problem that takes both the operational costs and precision of a policy into consideration.

The project is conducted at Philips. The interest of Philips is to reduce the maintenance costs of their healthcare systems by preventively replacing components, but only when a failure is imminent. The current predictive models use data from log files that are generated by the healthcare machines and we use the same data. The project focuses on the monitors of an iXR system, but the approach can be applied to other components as well.

The purpose of this report is to investigate how operational costs and a target value of a performance measure, such as the precision, can be taken into account, when obtaining a preventive maintenance policy.

Optimization problem The report focuses on the steps taken to derive an optimization problem to find a policy that balances the operational costs and precision measure.

The first step is to make a model to base our optimization on. Therefore, we introduce a stochastic model describing the degradation process of the component. To determine which features need to be described by the model, we rank the features used in a current machine learning model for the monitor based on their predictive value. The top three consist of two continuous features, the age and brightness, and one categorical feature. Historical data, obtained from log files, are used to estimate the parameters in the stochastic model.

The next step is to formulate the operational costs for a given policy. We consider two costs, namely a cost for corrective maintenance and a cost for preventive maintenance.It is difficult to obtain reliable estimates for the two costs and we therefore consider several ratios of the two costs. For Philips the precision of the replacements is important and we therefore propose two functions of the precision of a policy. Adding the operational costs and a function of the precision gives us the optimization problem.

Solutions The stochastic nature of the problem makes the optimization problem hard to solve. Therefore, we restrict our analysis by only considering three types of policies. The policies can be interpreted as curves that divide the positive quadrant of the plane spanned by age and brightness into two. The difference between the policies is the shape of the curve.

The first class of policies is based on thresholds for the age and brightness individually, resulting in a rectangle. The second class are ellipse shaped curves that correspond to the points at which the degradation process has a certain probability of being alive. The third class consists of linear planes, which include the classifiers obtained in the current machine learning model.

We cannot solve the optimization problem exactly and therefore we use approximations. For the first class, we look at the age and brightness thresholds individually. We can find both thresholds exactly and together they approximate the optimal policy in class 1. For the second and third classes, we use Monte Carlo simulation to estimate the objective function and to find the minimizers. The minimizers depend on the cost ratio and the relative importance given to the precision; a low weight results in a policy with many replacements and a high weight gives a conservative classifier with few replacements.

Results To compare the newly obtained policies with the old ones, we perform a simulation study. In the simulation we simulate the degradation process of monitors and apply the new and old policies. We see that the operational costs are significant lower for the proposed policies when the relative importance of the precision is low. For a cost ratio between the corrective and preventive replacements of 5, the decrease in cost is fourteen percent. When the ratio increases to 10, then the decrease is more than 28%. When more weight is put on the precision, then the performance of the new and old policies are similar.

Conclusion In this project, we see that including the operational costs in the decision-making process can lower the costs significantly. The price one pays for the lower costs is that more replacements are performed and that the replacements are performed earlier. This means that the lifetime of the components are used to a lesser extend.

The project is a start in incorporating operational costs in the decision-making process, but other factors, such as the impact of the increased number of replacements, need to be studied further.

Contents

List of Symbols

$\boldsymbol{\sigma}$	A (preventive) maintenance policy, see Policy.
$g, g(\boldsymbol{\sigma})$	The average cost and average cost of policy $\boldsymbol{\sigma}$.
$\ell, \ell(\boldsymbol{\sigma}, \alpha)$	Penalty function of the precision with target α when using policy $\boldsymbol{\sigma}$.
α	The target value of the precision.
k	The weight of the penalty term in the optimization problem.
BL	The base level of the monitors
$\boldsymbol{BB}$	The backlight brightness of the monitors as measured by the iXR systems.
$\boldsymbol{D}$	The degradation of a monitor, see also D.
X, X_i	The interarrival times of the jump process. We assume that the interarrival times are lognormal distributed.
Y, Y_i	The jumps of the jump process. We assume that the jumps are geometric distributed with mass function $\mathbb{P}(Y = k) = p(1-p)^{k-1}$, $k = 1, 2, \ldots$.
$D(t)$	The degradation process. Models the state of the monitor.
D	The degradation at failure.
L	The working hours at failure.
E	The time at which the environment switches states.
T, T_i	Moment of failure (in i-th cycle), see also Failure mechanism.
$S, S_i, S_{\boldsymbol{\sigma}}$	Moment that the policy $\boldsymbol{\sigma}$ prescribes a predictive replacement.
c_p, c_c	The cost of a preventive and corrective replacement. We assume that all costs regarding the replacement are captured by c_p and c_c.
R, R_i	The cost of a replacement, either c_c or c_p.
$N(t), M(t)$	Renewal process that count the number of arrivals until time t.
$V(t)$	The total costs made until time t, i.e. the sum of all costs until time t.
$h(t)$	The hazard rate.
$F(t), f(t)$	Cumulative density function and probability density function.

Definitions and Concepts

`TP, TN, FP, FN` The true positives, true negatives, false positives and false negatives of a binary classification.

Average cost The average cost is the total cost made in an interval divided by the length of the interval. In this project we let the length of the interval go to infinity.

BB The backlight brightness as measures by the monitors. The backlight brightness is one of the features in the predictive model that has the highest predictive power. We use the degradation in the degradation process, see $\boldsymbol{D}$.

BL The base level of the brightness of the monitors. The base level is the brightness level that the monitors should have according to specifications.

Censored data Censored data occurs when quantities are only partially known. In the report we deal with right censored data, where we only know that a certain quantity is larger than the observed value.

D The degradation of a monitor, defined as the difference between the base level BL and the backlight brightness $\boldsymbol{BB}$, i.e. $\boldsymbol{D} = BL - \boldsymbol{BB}$. See also $\boldsymbol{BB}$.

Degradation process The process that describes the state of a monitor. We assume that the process is a jump process.

Environment The environment models the internal state of the monitors and is binary. The environment takes values of 0 and 1, where a 1 indicates that something has happened and that the monitor is failing. We assume that there is enough time to perform a preventive replacements after the environment switches state.

Failure mechanism The failure mechanism describes how we assume the monitor to fail. In this report we assume that a monitor fails at time T, which is the first time that either the degradation crosses D, the working hours cross L or environment switches.

i.i.d. Independent and Identically Distributed random variable. Each random variable has the same probability distribution as the others and all are mutually independent.

Interarrival time The interarrival time is the time between two jumps of the degradation process. We assume that the jumps are continuous.

Jump process A jump process is a stochastic process that has discrete movements, called jumps, with random arrival times, called interarrival times.

Jump size The size of a jump in the degradation process. We assume that the jumps are discrete.

Operational costs The cost made when following a policy $\boldsymbol{\sigma}$. See also Average cost.

Policy A policy is a set of rules that prescribe when a component should be replaced. In this report we consider policies that divide the working hours-degradation plane into two: one part in which nothing is done and one part where a preventive replacement is performed. This means that a policy can be interpreted as a classifier, so we interchangeably use the terms policy and classifier when referring to $\boldsymbol{\sigma}$.

Precision The precision is the fraction of positive predicitions that are correct, i.e. $\texttt{TP}/(\texttt{TP}+\texttt{FP})$. The definition translates to the number of preventive replacements that occur in the predictive interval, which is defined as the 30 days before failure.

Predictive interval The interval of 30 days before a failure in which it is desired to have a preventive replacement.

Predictive model A model that predicts failures of a component. A predictive model consists of a set of rules that prescribe when a component should preventively be replaced to minimize a given objective, e.g. minimize operational costs.

Renewal process A counting process where the time between increments are i.i.d. distributed. The time between jumps are called interarrival times.

Running minimum The running minimum is used to model the real life data as a jump process. In the report we consider the third running minimum, which at time t is the third lowest value until time t.

SVM Support vector machines. A machine learning algorithm that finds a separating hyperplane between two (or more) classes on a training set and classifies new data points. An SVM classifier is currently used to predict failures.

WH The working hours is one of the features in the predictive model that has the highest predictive power. In the degradation process the working hours play the role of time.

Chapter 1

Introduction

Currently complex, highly valued healthcare machines are used in hospitals to improve the lives of countless people. Although the machines are designed to withstand the test of time, many components can, and will, fail at some point in time. Failures do not only cost money for all parties involved, but also endanger the safety of the customers.

To keep the machines running, maintenance is performed. The traditional approach is to replace machines when they fail. Currently, companies are transitioning towards *preventive maintenance* policies, where components are replaced when some condition is met. The goal is to decrease the downtime and maintenance costs of machines.

When a component is replaced after it failed, there is a long time that the machine cannot be used. The reason for this is that the failure must be reported, a new component must be shipped and the component must be replaced. On the other hand, when a component is preventively replaced, which can be done outside scheduled usage, the shipping is done while the machine is still working and valuable time is saved. A set of rules that describe when to replace a component is called a *policy*.

To determine the conditions when a component should be replaced, *predictive models* are developed. Currently, many predictive models are based on machine learning algorithms, such as *support vector machines* (SVMs). In 2014, Sipos et al. showed the usefulness of SVMs in predicting failures of medical systems [1]. However, the output of an SVM is a classifier that distinguishes between working components and failing components, which is not a policy. The classifier can be used as a policy, assigning an action to each of the two classes, but then the classifier is used in a way for which it was not developed. SVMs were not developed for sequential data and do not take costs attached to actions, taken based on the classification, into account (we discuss some recent developments treating these shortcomings in Chapter 9).

We therefore believe that improvements can be made when the costs attached to the actions are also included in the decision-making process. Models developed for sequential data and cost structures include *Markov decision processes* (MDPs), which have been used in many applications. The authors in [2] give a survey of papers treating real applications of MDPs. One of the applications categories in [2] is “Inspection, maintenance and repair”, which discuss problems related to preventively replacing machines.

1.1 Main Optimization Problem

In this report we discuss one approach to extend MDPs to also take a performance measure, similar to the ones in machine learning, into account. We thus look at two performance indicators when evaluating a (preventive) policy $\boldsymbol{\sigma}$. The first indicator is the *operational costs*, i.e. the costs when following a policy $\boldsymbol{\sigma}$, denoted by $g(\boldsymbol{\sigma})$. The second indicator is a function of the *precision*, which measures the fraction of replacements that were performed close to the actual failure, i.e. the moment the machine fails when no preventive maintenance is performed, and a target value α of the precision. We denote the second performance indicator by $\ell(\boldsymbol{\sigma}, \alpha)$. The work results in the following optimization problem

$$\operatorname*{arg\,min}_{\boldsymbol{\sigma} \in \mathcal{S}} \Big\{ g(\boldsymbol{\sigma}) + \ell(\boldsymbol{\sigma}, \alpha) \Big\}, \tag{1.1}$$

where $\mathcal{S}$ is the set of all policies $\boldsymbol{\sigma}$. In Equation (1.1) a trade-off between the operational costs and precision with target α is found. All quantities in Equation (1.1) are stochastic and are based on a stochastic process that describes the state of the component.

1.2 Contribution of Project

The main novelty in this project is the optimization problem that finds a trade-off between two performance indicators, namely the cost and the precision. Both machine learning methods and MDPs focus on one performance indicator, but not both. Combining both gives new insights on the relation of the two and what decision companies need to make regarding the characteristics of their maintenance policies.

Other differences with previous work include the usage of an existing machine learning model. We use the model to choose the variables that the stochastic process models. The choice is based on a ranking of the features on their predictive power. Furthermore, unlike MDPs we do not assume that certain processes follow the exponential distribution, which gives us more flexibility to model the degradation process of the monitors.

1.3 Outline

The outline of the thesis is as follows. In the second chapter we introduce the big screen monitors, which we use as vehicle of illustration in the report. In Chapter 3 we introduce the stochastic process that is used to describe the state of a monitor. Chapter 4 discusses three classes of policies that we consider in the report. In Chapter 5 we introduce the average cost criterion, which we use to calculate the operational costs. Next, in Chapter 6 we discuss how the precision is accounted for in the optimization problem. The complete formulation and approaches to solve the optimization problem are shown in Chapter 7. In Chapter 8 we compare the proposed policies with the old policy via a simulation study. In Chapter 9 we discuss some recent literature on SVMs that include one of the performance indicators in the SVM formulation and the problems in applying the results to our case. We end the report with a conclusion and discussion in Chapter 10.

Chapter 2

Vehicle of Illustration: Big Screen Monitors

To demonstrate the steps that one needs to take to obtain the main optimization problem in Equation (1.1) and to compare the performance of the solution to the current situation, we use a specific component of the iXR systems. The component that we consider is the big screen monitor, which we introduce in this chapter.

We first discuss the usage of the big screen monitors. After that, we look at the current machine learning implementation to predict failures of the monitors. Then we rank the features of the predictive model based on their predictive power. In the next chapter, we use the ranking to develop a stochastic process that models the state of the monitor based on the three features that topped the ranking. Therefore, we end the chapter with a brief discussion on the three topped ranked features.

2.1 The Big Screen Monitors

In this project we use the big screen monitors, see Figure 2.1, as a vehicle of illustration. The monitors are also called FlexVision monitors, since the monitors can remember the preferences of different doctors. The preferences determine what information is shown where on the screen during operations, hence FlexVision (flexible vision).

The big screens are a component on iXR systems that are used for image-guided therapy, where images are used to help doctors perform surgical procedures and therapeutic interventions. One big advantage of image-guided therapy is that it is less invasive than a surgery, reducing the pain, recovery time and physical trauma[1].

Each iXR machine generates a lot of data while in use. The data is stored in log files, which are send regularly to Philips. The predictive models use the data from the log files to make predictions. [redacted]

[1]See https://www.youtube.com/watch?v=pZg1Gg0tJPw for a live operation using image-guided therapy with a Philips medical device.

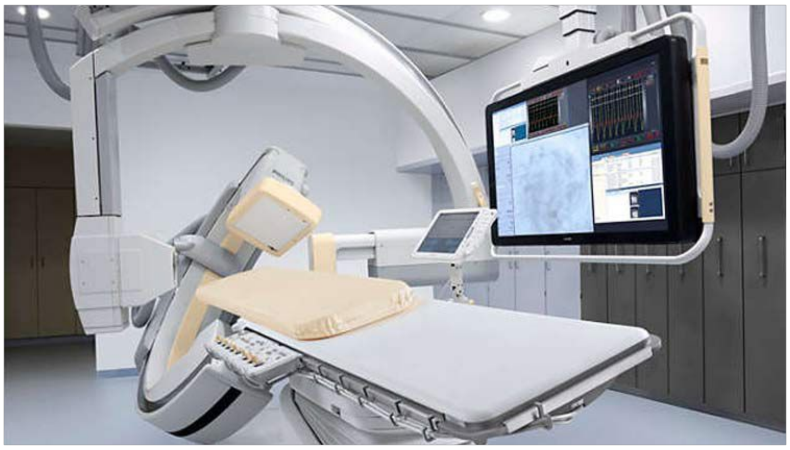

Figure 2.1: Photo of the Allura Xper FD20/10 X-ray system with the big screen in the top right corner. Source `https://www.philips.co.uk/healthcare/product/HC722029CA/allura-xper-fd20-10-biplane-cardiovascular-x-ray-system`[accessed 26-02-2019].

Table 2.1: [redacted]

[redacted]

2.2 The SVM Model

To predict failures of the monitors, a linear SVM model is used. The model is trained on a training set, which consists of eleven features and labels corresponding to the entries. The features can be divided in groups, including, for example, the ones connected to the physical appearance of the image (contrast etc.), the ones related to the specific use of the monitor (like energy used) and more general ones connected to the internal workings of the hardware itself.

The input of the model are the eleven dimensional input vectors $\boldsymbol{x}$ containing the values of the eleven features and the labels y, set to -1 and $+1$, corresponding to all the input vectors. The output of a linear SVM is a hyperplane of the form $\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = 0$, where $\boldsymbol{w}$ is an eleven dimensional weight vector and b the bias or intercept, and is obtained by solving the following optimization problem

$$\begin{aligned} &\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{n} \xi_i \\ &s.t.\ y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \dots, n. \end{aligned} \tag{2.1}$$

To guarantee a solution, the formulation in Equation (2.1) includes a penalty C for all wrongly classified data points and a slack ξ, larger that zero, that accounts for the amount of violation of the misclassified points.

The class of a new point $\boldsymbol{x}_{\text{new}}$ is determined by the sign of $\langle \boldsymbol{w}, \boldsymbol{x}_{\text{new}} \rangle + b$. The outcome of the SVM model is thus binary; a new data point is classified as working (-1) or failing $(+1)$. In the latter case the data of the monitor is inspected in more detail to determine the following actions, e.g. a possible preventive replacement. For more information on SVMs, see, for instance, [3] and [4].

[redacted]

2.3 Feature Ranking

We want to develop a stochastic process that accurately models the state of a monitor without becoming too complex with respect to the decision, since the complexity hugely impacts the optimization complexity. To achieve an accurate and relative simple model, we only want to use a few variables that give a lot of information about the state of a monitor.

To find variables for the model, we turn our attention to the machine learning model that in use. The current SVM model has eleven features; using all feature would make the process too complex. However, not all features contribute evenly to the preventive maintenance decision. Therefore, ranking the features on their predictive power allows us to select the features that give the most information about the state of a monitor. Note that the features are thus not chosen to fit the data, but to describe the patterns that reflect the decision in the SVM model.

One way to rank the features in a linear SVM model is by the weight vector. We know that the class of a new point $\boldsymbol{x}_{\text{new}}$ is determined by the sign of $\langle \boldsymbol{w}, \boldsymbol{x}_{\text{new}} \rangle + b$. An absolute higher value of a component in the weight vector means that any change in the measurement of the corresponding feature has a higher impact on the value of the dot product than a similar change of a component with an absolute lower weight. We thus interpret the features with higher weights as the ones having a high predictive power. The components of the weight vector of the model are shown in Table 2.2a.

The SHAP values give us a second way to rank the features. In [6] the authors introduce the SHAP values, which assign an importance value to each feature. The theory is inspired by a concept of cooperative game theory, namely Shapley values (see, for instance, [7] for an introduction to game theory and Shapley values, in particular Section 9.4). The SHAP values are calculated with the SVM model and explain how the features contribute to the classification of a new point.

Just like the Shapley values, there are three desired properties that uniquely define the SHAP values. Before we state the properties, we introduce the notation used in [6]. Let f be the original model, which we want to explain, and let g be the explanation model. The explanation model uses simplified inputs $x' \in \{0,1\}^M$, where M is the number of features of the model. The original inputs x are obtained by a mapping h_x that converts a binary vector of interpretable inputs into the original input space, i.e. $x = h_x(x')$. Note that h_x depends on x.

The three properties that define the SHAP value are (paraphrased from [6])

Property 1 (Local accuracy) The explanation model $g(x')$ matches the original model $f(x)$ when $x = h_x(x')$ and is a linear function with real coefficients ϕ_i for $i = 0, 1, \dots, M$, i.e.

$$f(x) = g(x') = \phi_0 + \sum_{i=1}^{M} \phi_i x'_i.$$

The coefficient ϕ_i is called the attribution of feature i.

Property 2 (Missingness) The missingness property assures that features that are not included do not have an attributed impact, so $x'_i = 0 \Rightarrow \phi_i = 0$.

Property 3 (Consistency) If the model changes such that the contribution of some simplified input increases or stays the same regardless of the other inputs, then the input's attribution should also stay the same or increase. Let $z' \setminus i$ denote the setting $z'_i = 0$ and let $f_x(z') = f(h_x(z')$ for $z' \in \{0,1\}^M$. For any two models f and f', if

$$f'_x(z') - f'_x(z' \setminus i) \geq f_x(z') - f_x(z' \setminus i)$$

for all inputs $z' \in \{0,1\}^M$, then $\phi_i(f', x) \geq \phi_i(f, x)$.

The SHAP values are given by Theorem 2.1.

Theorem 2.1. *[Theorem 1 of [6]] There is only one possible explanation model g that satisfies properties 1, 2 and 3 and the features attributions are given by*

$$\phi_i(f,x) = \sum_{z' \subseteq x'} \frac{|z'|!(M - |z'| - 1)!}{M!}[f_x(z') - f_x(z' \setminus i)],$$

where $|z'|$ is number of non-zero entries in z' and $z' \subseteq x$' represents all vectors z' where the non-zero entries are a subset of the non-zero entries in x'

Table 2.2: The eleven features of the SVM model ordered on their contribution to new classifications.

(a) Weights of the SVM hyperplane

Feature	Weight
Feature 2	$-2.473 \cdot 10^{-1}$
Feature 5	$1.742 \cdot 10^{-1}$
Feature 1	$-1.737 \cdot 10^{-1}$
Feature 6	$1.290 \cdot 10^{-1}$
Feature 4	$6.770 \cdot 10^{-2}$
Feature 3	$6.432 \cdot 10^{-2}$
Feature 7	$-4.105 \cdot 10^{-2}$
Feature 8	$-3.891 \cdot 10^{-2}$
Feature 9	$-2.255 \cdot 10^{-2}$
Feature 10	$3.074 \cdot 10^{-18}$
Feature 11	$3.074 \cdot 10^{-18}$

(b) Averaged Shap values.

Feature	SHAP
Feature 5	$2.562 \cdot 10^{-2}$
Feature 2	$2.380 \cdot 10^{-2}$
Feature 1	$6.770 \cdot 10^{-3}$
Feature 6	$6.033 \cdot 10^{-3}$
Feature 3	$3.223 \cdot 10^{-3}$
Feature 4	$2.258 \cdot 10^{-3}$
Feature 7	$2.041 \cdot 10^{-3}$
Feature 8	$4.864 \cdot 10^{-4}$
Feature 9	$3.939 \cdot 10^{-5}$
Feature 10	0.
Feature 11	0.

Unlike the weight vector, the SHAP values are calculated per prediction and not for the whole model. To obtain a ranking of the features based on the SHAP values, we divide the training set into 10 equally sized groups using stratified sampling and use 10-fold cross-validation. We use stratified sampling to make sure that the ratio between working and failed monitors is constant over the groups and choose 10-fold cross-validation since it common practice and it was used in the original training of the SVM model. We then train an SVM model on nine of the groups and aggregate the absolute values of the SHAP values of each prediction, see Algorithm 2.1. To train the SVM classifier we use the R function `svm()` in the `e1071` package [5] and the same parameter values as the original model (see Section 2.2). We use the `iml` [8] package in R to calculate the SHAP values and, apart from the sample size, which is increased to 500, use the default settings of the function `Predictor$new()`. The arithmetic mean of the SHAP values per feature are shown in Table 2.2b.

The implementation in the `iml` package calculates sampling-based estimates of the SHAP values, which is discussed in detail in [9]. The idea is to use an alternative formulation of the Shapley value that uses the ordered list of permutations of the features. Due to the sampling, the results differ slightly between runs. However, we noticed that the same conclusion, which we discuss later in this section, can be drawn from any of the runs, so the method gives consistent results.

Algorithm 2.1: Procedure to calculate the SHAP values of each feature of the SVM model using stratified sampling and cross-validation.

```
   Input: Training set (TS) for big screens
   Output: Arithmetic mean of the SHAP value per feature of the instances in the
           training set
   // Step one:Splitting
 1 Make sets Fail and Normal consisting of row indices in TS corresponding to
    these labels
 2 Set $n_F = |Fail|/10$ and $n_N = |Normal|/10$
 3 for $i$ in 1 to 10 do
 4     sample $n_F$ indices from Fail
 5     sample $n_N$ indices from Normal
 6     set $i$ is union of both samples
 7     remove sampled indices from the sets Fail and Normal
 8 Distributed possible remaining indices over sets
   // Step two:  Determine SHAP values
 9 Initialize matrix SH.tot for the SHAP values per feature
10 for $i$ in 1 to 10 do
11     Test set is the subset of TS consisting of indices in set $i$
12     Training set (TN) is the rest of TS
13     Train an SVM on training set
14     for $\boldsymbol{x}$ in TN do
15         Compute vector SH with SHAP values of $\boldsymbol{x}$
16         SH.tot[$i$, ] = SH.tot[$i$, ] + SH            // row $i$ in matrix SH.tot
17 Calculate the arithmetic mean per feature (column) to obtain the mean SHAP
    value per feature.
```

A third way to rank the features is by using a forward selection procedure, based on to the forward selection procedure in stepwise regression (see, for instance, Section 9.12.2 in [10]). The idea is to start with zero features and add one feature at the time until all eleven features are included. In each step we train one SVM model for each feature not yet selected where the features in the model are the ones already selected features and the new one. The feature selected in a step is the feature that improves some classification measure the most, see Algorithm 2.2.

The classification measure is based on the confusion matrix, see, for instance, [11], that gives the number of true positives, true negative, false positives and false negatives of a classification. The SVM model classifies each input vector in the training set as either positive or negative and each classification thus falls in one of the four categories.

Algorithm 2.2: Procedure of the forward feature selection to rank features on their predictive power.

```
Input: Training set (TS) for big screens, measure $M$
Output: List of features selected by forward selection.
1  Split TS into a training set and test set.
2  Set $AF = \{1, 2, \dots, 11\}$              // vector with all feature labels
3  Set $SF = \emptyset$                      // vector with the selected features
4  while $AF \neq \emptyset$ do
5      Initialize $per$, constant vector of length 11 with value $-10$
6      for $i$ in $AF$ do
7          Train SVM with features $SF \cup \{i\}$ on train set
8          Calculate performance on test set according to measure $M$ and store in
             per[i]
9      $k = \arg\max per$                    // feature with the best performance
10     $SF = SF \cup \{k\}$
11     $AF = AF \setminus \{k\}$
```

As classification measure, we use the precision measure for reasons that we discuss in Chapter 6. The precision is defined as "the number of correctly classified positive examples divided by the number of examples labeled by the system as positive" [11], i.e.

$$precision = \frac{\mathtt{TP}}{\mathtt{TP} + \mathtt{FP}}$$

The results of the forward selection procedure are shown in Table 2.3. In this particular case, the order of inclusion seems arbitrary for several features, since they do not add anything to the precision. However, including any of the bottom six features before any of the top five does result in a slower increase, i.e. the maximum precision is reached with more features. The order in Table 2.3 is thus consistent with the results of Table 2.2.

We see that the first two methods rank the features in almost the same order, only feature 2 and feature 5, and feature 3 and feature 4 switch position. Both methods indicate that features 1, 2 and 5 are the features that contribute the most. The results of the third method support the findings of the first two. The found order importance corresponds to the opinion of domain experts at Philips.

2.4 Brief Description of Top Ranked Features

The goal of the next chapter is to develop a stochastic process that models the state of a monitor. We want this model to be accurate, but not too complex. We therefore choose to use the top three ranked features, which we briefly discuss in this section.

[redacted]

Table 2.3: Features as selected by the forward selection procedure based on the precision measure. Even though Features 1 and 2 do not immediately increase the precision, they must be included in this order to obtain the maximum precision with as few features as possible.

Feature	Increase	Total
Feature 5	$6.833 \cdot 10^{-1}$	0.6833
Feature 2	0.	0.6833
Feature 1	0.	0.6833
Feature 3	0.	0.6833
Feature 4	0.	0.6833
Feature 9	$4.749 \cdot 10^{-3}$	0.6880
Feature 6	$8.915 \cdot 10^{-3}$	0.6969
Feature 7	0.	0.6969
Feature 8	0.	0.6969
Feature 10	0.	0.6969
Feature 11	0.	0.6969

[redacted]

Chapter 3

Degradation process

In this chapter we introduce the stochastic process that we use to model the state of a monitor. In Chapter 2 we discuss the current SVM implementation to predict failures of the monitors. We end the chapter by discussing the three features with the most predictive power.

In this chapter we first use the three features to formulate a stochastic process, called the degradation process, to model the state of a monitor. We then state the failure mechanism that we assume and look at the assumptions that we make in the formulation and whether they are satisfied or not. We end the chapter with estimations for the stochastic quantities of the degradation process, which we obtain by using real world data of the big screen monitors.

3.1 The Degradation Process

We can capture two of the top three ranked features, namely the working hours and brightness, by following the brightness over time. However, instead of looking at the measured values of the brightness, we look at the degradation from the base level. We define the *degradation* $\boldsymbol{D}$ as the difference between the base level and the measured value of the brightness, i.e.

$$\boldsymbol{D} = BL - \boldsymbol{BB}. \tag{3.1}$$

In Equation (3.1) $\boldsymbol{BB}$ is the backlight brightness as measured by the monitors and BL is the base level.

In Equation (3.1), we see that it is mathematically equivalent to look at the brightness or the degradation. However, there are several advantages of looking at the degradation instead of the brightness directly. One advantage is that we can model the *degradation process*, i.e. the degradation of a monitor over time, as a non-decreasing sequence starting at zero, which allows us to drop the base level in most of the calculations.

The recorded brightness measurements vary a lot, see Figure 3.1. Following the measurements over time would thus give us a volatile process. To obtain a stable process, we define an envelope that follows the lower part of the measurements. The assumption behind the usage of the envelope is that the lower values give more information about the "real" value of the brightness. The envelope should accurately follow the lower measurements, but still be robust to some outliers, for instance, brightness measurements of zero. An envelope that satisfies these constraints is the third running minimum. The definition of the third *running minimum* at time t is the third lowest point until time t. In Section 3.3 we discuss why we choose the third running minimum as envelope.

The brightness of a monitor starts at the base level and then decreases over time. The degradation thus starts at zero and increases over time. We assume that, when the brightness decreases, i.e. the third running minimum changes value, the brightness decreases by an integer valued random variable and that the times between decreases are continuous. We call the decreases *jumps* and the time between jumps *interarrival times*. Furthermore, we assume that both the jumps and interarrival times are independent and identically distributed (i.i.d.) and that the jumps and interarrival times do not depend on each other, i.e. they are independent. Both the jumps and the interarrival times are positive random variables. We denote the i-th jump and interarrival time by Y_i and X_i, respectively.

The assumptions allow us to model the degradation process as a *jump process* $D(t)$, given by

$$D(t) = \sum_{i=1}^{N(t)} Y_i. \tag{3.2}$$

In Equation (3.2), $N(t)$ is the number of jumps until time t, which is given by

$$N(t) = \left| \left\{ n \in \mathbb{N}_+ \mid \sum_{i=1}^{n} X_i \leq t \right\} \right| = \max \left\{ n \in \mathbb{N}_+ \mid \sum_{i=1}^{n} X_i \leq t \right\}. \tag{3.3}$$

The third feature that has high predictive value in the SVM classifier is a feature connected to the internal workings of the hardware (feature 5). The feature takes the values 0 and 1, where a 1 indicates that something undesirable has happened and the monitor is about to fail. We model this feature as an environmental factor which takes values 0 and 1. The time it takes the environment to switch from 0 to 1, denoted by E, follows some distribution that is independent of the degradation process and has support on the positive reals. We denote the environment at time t by $Env(t)$. Finally, we assume that there is enough time to schedule a preventive maintenance visit after the environment switches.

3.2 Failure Mechanism

We assume that a monitor fails if one of three events occur: the working hours cross some level, the brightness drops below some level (the degradation thus crosses a level) or the environment switches. The values that the working hours and brightness need to cross to cause a failure are random variables, which we denote by L and B, respectively (see Figure 3.1). We define D as the amount of degradation needed to fail, which is related to B according to Equation (3.1). We assume that L, similar to E, has support on the positive reals and D on the closed interval $[0, BL]$. Finally, we assume that L, D and E

Figure 3.1: The measured brightness values over time with the third running minimum $B(t)$ and the values at failure of the brightness and working hours.

are independent. The time at which a monitor fails is thus given by

$$T = \inf\{t \mid t \in \mathbb{R}, \mathbb{1}\{\{t \geq L\} \cup \{t \geq E\} \cup \{D(t) \geq D\}\} = 1\}. \tag{3.4}$$

There is one physical constraint in the model, namely the degradation must be below the value of the base level. If this is not the case, then, by Equation (3.1), the brightness is negative, which is impossible. However, the condition is never violated, since the definition in Equation (3.4) ensures that the failure time occurs before the brightness reaches 0.

3.3 Validation of Assumptions

In Section 3.1 we make some assumptions on the variables in the degradation process. In this section we briefly discuss why the assumptions are reasonable.

The main assumption that we make is that the degradation process can be modeled as a jump process. In the top graph of Figure 3.2, we plot the measured brightness values for a single monitor over time (black dots) and the third running minimum (blue line), which represents the jump process. We use the third minimum, because it is less sensitive to single measurements than the minimum and still flexible enough to follow changes in the measurements. The third minimum seems to follow the lower part of the measurements well, which also holds in general, and is by construction a jump process. The degradation process, obtained by subtracting the third minimum from the base level (see Equation (3.1)), is thus also a jump process, see the red line in the bottom graph of Figure 3.2.

Figure 3.2: The measured brightness values over time with the third running minimum (blue line) and the corresponding degradation process $D(t)$ (red line) for a monitor of an iXR sytsem.

The other big assumption we make is that the interarrival times and jump sizes are i.i.d. and independent of each other. To test the assumed independence, we use data from monitors that are included in the training set used to train the SVM classifier (for the rest of the chapter we refer to this set by training set). For each of the 518 unique monitors in the training set, we determine the third running minimum to calculate the interarrival times and jump sizes. To test the assumptions we perform three tests on both the interarrival times and the jumps, namely the Wald-Wolfowitz runs test to test the randomness of the data [12], the Ljung-Box test to check whether the data is independently distributed [13, 14] and the difference-sign test, which tests the i.i.d. assumption of the data [14]. The Wald-Wolfowitz and difference-sign tests are implemented in the R package `randtest` [15] through the functions `runs.test()` and `difference.sign.test()`, respectively. The Ljung-Box test is performed using the `Box.test()` function in R. In Appendix B the hypothesis and test statistics are shown in detail. The testing procedure is outlined in Algorithm 3.1.

Algorithm 3.1: Procedure to calculate the fraction of null hypotheses that are not rejected for several tests on the randomness and independence of the interarrival times and jump sizes of the jump process. The results are shown in Table 3.1.

```
   Input: Set of monitors ($M$) and their $\boldsymbol{WH}$ and $\boldsymbol{BB}$ measurements over time
   Output: Fraction of monitors that reject the null hypothesis for each test
1  for $m$ in $M$ do
2      Construct third running minimum of $m$ from $\boldsymbol{WH}$ and $\boldsymbol{BB}$ measurements
3      Let $n.jumps$ be the number of jumps in the third minimum
4      if $n.jumps > 4$ then
           // Compute tests on both interarrival times and jumps
5          Perform Wald-Wolfowitz test and store p-value
6          Perform Ljung-Box test and store p-value
7          Perform difference-sign test and store p-value
8  Compute fraction of null hypotheses that are not rejected at a 95% confidence
     level
```

In Table 3.1 the results of the three test are shown. We only consider monitors for which the third minimum has at least five jumps, which leaves us with 494 monitors. The values in the table represent the fraction of monitors that did not reject the null hypothesis at a 95% confidence level. For instance, 80 percent of the monitors have a sequence of interarrival times for which the Wald-Wolfowitz does not have enough evidence to reject the randomness hypothesis. We see that for most monitors the i.i.d. assumption on the interarrival times and jump sizes are reasonable.

Table 3.1: The fraction of monitors that did not reject the null hypotheses for the independence tests at a 95% confidence level for the interarrival times and jump sizes per monitor.

Test	**Fraction not rejected**	
	Interarrival	Jumps
Wald-Wolfowitz	0.7996	0.8798
Ljung-Box	0.9271	0.8684
difference-sign	0.9271	0.9737

The last part of the assumption is that the interarrival times and jump sizes are independent form each other. To test whether there is a relationship between the interarrival times and the jump sizes, we use a set of four tests introduced by Heller, Heller and Gorfine [16]. The null hypothesis for the tests are that two vectors, the interarrival times and jump sizes, are independent; the alternative is that the two vectors are dependent. The tests are implemented in the package `HHG` [17] in R. Since the tests are based on pairwise distances between the samples of the two vectors, we scale the vectors to zero mean and unit variance and consider the Euclidean distance. We use a similar testing procedure as represented in Algorithm 3.1. The difference is that we now use the standardized working hours and brightness measurements and use the `hhg.test()` function.

The results of the tests are shown in Table 3.2. The fraction of null hypotheses that is not rejected is similar to the non rejection rates in Table 3.1. Therefore, we draw a conclusion along the same lines and say that independence between interarrival times and jump sizes seems reasonable.

Table 3.2: The fraction of monitors that did not reject the null hypotheses for the HHG tests at a 95% confidence level for the independence of the interarrival times and jump sizes.

Test Variant	**Fraction not rejected**
sum chi square	0.8619
sum likelihood	0.8619
max chi square	0.9095
max likelihood	0.8786

The final assumption is related to the failure mechanism of the monitors, namely we assume that the value of the working hours and degradation at failure are i.i.d. and independent of each other. The data available to test the assumptions are the failure instances in the training set. Using the working hours and brightness at failure and applying the same tests as for the interarrival times and jumps, we obtain p-values that are all larger than 0.05. The assumptions thus seem reasonable.

3.4 Estimations of Parameters

The final step in setting up the model is to specify the distributions of the different stochastic quantities. We first discuss the distributions for the jump process, i.e. the distributions for the jumps and interarrival times, and then look at the values at failure.

3.4.1 Jump Process

The specification of the jump process is done in three steps: first the parameters of several distributions per monitor are estimated, then the hypothesis that the data comes from each of the distributions is tested and finally the parameters are selected.

To estimate the parameter of a distribution, we use *maximum likelihood estimation.* Assume that $\boldsymbol{X} = (X_1, X_2, \ldots, X_n)$ are i.i.d. random variables, each with density $f_{\boldsymbol{\theta}}(x)$, where $\boldsymbol{\theta}$ is the vector with the parameters of the distribution, and let $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ be a realization of $\boldsymbol{X}$. The likelihood function $L(\boldsymbol{x}, \boldsymbol{\theta})$ is the probability of the realization $\boldsymbol{x}$ for parameter vector $\boldsymbol{\theta}$. We have that

$$L(\boldsymbol{x}, \boldsymbol{\theta}) = \prod_{i=1}^{n} f_\theta(x_i). \tag{3.5}$$

The idea of the maximum likelihood estimation is to find the value of $\boldsymbol{\theta}$ that maximizes the probability of realization $\boldsymbol{x}$, i.e. maximizing Equation (3.5) [18]. Often the log-likelihood is considered, because the log transformation simplifies calculations.

The data for the jumps fits nicely in the above framework for maximum likelihood estimation. However, the data for the interarrival times and the values at failure in the training set do not. For instance, the value of the last interarrival time is not known, since we do not know when that jump takes place. We only know that the interarrival time is at least as big as the time since the last jump. This means that the value in which we are interested, is only partially known. Data that is only partially known is called censored data. When the unknown part of the data is at the end, we have *right-censored data.*

The maximum likelihood estimation procedure can be adapted to censored data, see, for instance, Chapter 2 of [19]. To this end, we assume that we have right-censored data and that we have a binary variables $\boldsymbol{r} = (r_1, r_2, \ldots, r_n)$ indicting whether realization x_i is fully known ($+1$) or censored (-1). The likelihood function for a distribution F with density function f and parameter vector $\boldsymbol{\theta}$ is then given by

$$L(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) = \prod_{i=1}^{n} f(x_i, \boldsymbol{\theta})^{\frac{r_i+1}{2}} [1 - F(x_i, \boldsymbol{\theta})]^{\frac{1-r_i}{2}} = \prod_{i\,:\,r_i=+1} f(x_i, \boldsymbol{\theta}) \prod_{i\,:\,r_i=-1} 1 - F(x_i, \boldsymbol{\theta}). \tag{3.6}$$

Note that Equation (3.5) is a special case of Equation (3.6), which we obtain when $r_i = 1$ for $i = 1, 2, \ldots, n$ and use the fact that the empty product equals 1.

To test whether the data could be a sample from the distribution with the estimated parameters, i.e. the maximizers of Equation (3.6), we want to perform a statistical test with null hypothesis that the data is a sample from the specified distribution. However, since we used the data to estimate the parameters of the distribution we want to test against, the classical Kolmogorov–Smirnov test becomes biased towards the null hypothesis; the result is that the test is too conservative. Therefore, we use a parametric bootstrap procedure described in Section 2 of [20] to estimate the p-value of the hypothesis. The test statistic used in the bootstrap procedure is the Cramér-von

Algorithm 3.2: Procedure to obtain the fraction of monitors that do not reject the hypothesis that the data is a sample of a certain distributions. We use a parametric bootstrap procedure to approximate the p-value. As example, we use the interarrival time. We set n_B to 1000 in the calculations.

```
Input: List of monitors (M), set of distributions (Dists) and $n_B$ the number of
       bootstrap samples.
Output: Parameter estimates for distribution at failure of several distributions.
1  for $m$ in $M$ do
2      Construct third running minimum of $m$
3      Let $n$ be the number of recorded interarrival times
4      Let $x_1, x_2, \dots, x_n$ be the interarrival times
5      Calculate the order statistics $x_{(1)}, x_{(2)}, \dots, x_{(n)}$
6      if $n$ larger than 4 then
7          for $F$ in Dists do
8              Calculate CvM test statistic of F, $T_F := \frac{1}{12n} + \sum_{i=1}^{n} \left[\frac{2i-1}{2n} - F(x_{(i)})\right]^2$
9              Estimate the parameters $\hat{\boldsymbol{\theta}}$ of $F$ by maximum likelihood estimation,
                i.e. maximizing Equation (3.6)
10             for $i$ in 1 to $n_B$ do
11                 Draw $n$ random variables $\widetilde{x}_1, \widetilde{x}_2, \dots, \widetilde{x}_n$ from $F$ with parameters $\hat{\boldsymbol{\theta}}$
12                 Calculate the order statistics $\widetilde{x}_{(1)}, \widetilde{x}_{(2)}, \dots, \widetilde{x}_{(n)}$
13                 Calculate CvM test statist $T_i := \frac{1}{12n} + \sum_{i=1}^{n} \left[\frac{2i-1}{2n} - F(\widetilde{x}_{(i)})\right]^2$
14             Compute bootstrap p-value, given by $p = \frac{1}{n_B} \sum_{i=1}^{n_B} \mathbb{1}\{T_i > T_F\}$
15 Compute fraction of monitors that do not reject null hypothesis at confidence
     level $\alpha$
```

Mises (CvM) test statistic T, given by

$$T_F := \frac{1}{12n} + \sum_{i=1}^{n} \left[\frac{2i-1}{2n} - F(x_{(i)})\right]^2,$$

where F is the distribution we want to test against and $x_{(i)}$ for $i = 1, 2 \dots, n$ are the order statistics of the realizations x_i. The full procedure, together with the parameter estimation per monitor, is shown in Algorithm 3.2.

We use 1000 bootstrap samples for each monitor that has at least five interarrival times and test four distributions that have positive support, namely the exponential, Weibull, lognormal and gamma distributions. To find the maximizers of the likelihood function we use the functions `fitdist()` (non-censored data) and `fitdistcens()` (censored data) from the `fitdistrplus` package [21] in R.

Table 3.3 shows the fraction of monitors that did not reject the null hypothesis that the interarrival times came from the tested distribution. The highest fraction in Table 3.3

comes from the lognormal distribution and we therefore choose to model the interarrival times as coming from a lognormal distribution.

Table 3.3: Fraction of monitors that do not reject the null hypothesis that the interarrival times come for the given distribution at confidence level 0.05. The individual bootstrap p-values are obtained using 1000 bootstrap samples.

Distribution	**Fraction not rejected**
Exponential	0.1964
Weibull	0.5486
Lognormal	0.7368
Gamma	0.6235

We follow the same procedure for the jump sizes. All jumps are integer valued, so we test some discrete distributions. We test the (shifted) geometric distribution, which has probability mass function $\mathbb{P}(X = k) = p(1-p)^{k-1}$ for $k = 1, 2, \ldots$ and $p \in (0, 1)$, the Poisson distribution and the negative binomial distribution. The results are shown in Table 3.4, where we again considered monitors with at least five jumps. We see that the geometric distribution is a good fit for many monitors and therefore choose to model the jump sizes as coming from a geometric distribution.

Table 3.4: Fraction of monitors that do not reject the null hypothesis that the jumps sizes come for the given distribution at confidence level 0.05. The individual bootstrap p-values are obtained using 1000 bootstrap samples.

Distribution	**Fraction not rejected**
Geometric	0.7237
Poisson	0.3026
Negative Binomial	0.2500

The parameter estimations for the lognormal distribution fall in a wide range, see Appendix A. The wide range of estimations indicate that the degradation process is monitor dependent. Similarly, the estimated parameter for the jump size distribution also varies a lot and there seems to be no relation between the estimates for the interarrival time and jump sizes.

For this project we select a monitor that we believe is representative for the whole group. The parameter values of this monitor are $\mu = 4.198$, $\sigma = 1.401$ and $p = 0.6078$. The majority of monitors do not reject these parameters as the true parameters when using the bootstrapping method. For the remainder of the report we therefore assume these parameters to be the true parameters for all monitors, which might be a too strict assumption.

In future work one could replace the deterministic parameters by random quantities. Having random parameters allows the model to be more representative for all monitors.

If one uses the model for decision-making, then one can update the parameters of a specific monitor as time goes by. One possibility is to use Bayesian updating for such a model.

3.4.2 Values at Failure

In this section we discuss how the distributions and parameters for the working hours at failure (L), the degradation at failure (D) and time until the environment switches (E) are selected. In this section we use an adapted form of the training set, where all instances of the same monitors have been averaged out. The reason for this is that multiple instances of the failures are included in the training set, which are all labeled as failing. These instances make the data dependent, which would not allow us to use the maximum likelihood estimation procedure.

One problem with the data is that the reason why a monitor failed is not given. Since we currently do not know a way to obtain the reason of failure, we use all the failures as uncensored data for each estimation. Essentially, we combine censored and uncensored data and therefore make a mistake in the estimations of the values at failure.

A possible solution to the problem is to collect the reasons of failure as well, which would allow us to use the maximum likelihood estimation for right-censored data, see Section 3.4.1). The extra data would also benefit other predictive models. A point of future research is to develop techniques to better estimate distributions in case like this.

Working Hours Previous research (see [22]) identifies some candidate distributions to model the working hours at failure. The proposed distributions are the exponential distribution, the Weibull distribution, the lognormal distribution, Lai et al.'s modified Weibull distribution [23], the Hjorth distribution [24], and the Xie and Lai's modified Weibull distribution [25]. In Table 3.5 the cumulative distributions function of the six aforementioned distributions are shown.

Table 3.5: The distribution functions of the distributions under consideration for the working hours at failure.

Distribution	$\boldsymbol{F(x)}$	**Parameters**
Exponential	$1 - \mathrm{e}^{-\lambda x}$	$\lambda > 0$
Weibull	$1 - \mathrm{e}^{-(x/\lambda)^k}$	$\lambda, k > 0$
Lognormal	$\frac{1}{2} + \frac{1}{2}\,\mathrm{erf}\left[\frac{\ln x - \mu}{\sqrt{2}\sigma}\right]$	$\sigma > 0, \mu \in (-\infty, \infty)$
Lai et al.[23]	$1 - \mathrm{e}^{-\lambda x^{\beta} \mathrm{e}^{\nu x}}$	$\lambda, \beta, \nu > 0$
Hjorth [24]	$1 - \frac{\mathrm{e}^{-\delta x^2/2}}{(1+\beta x)^{\theta/\beta}}$	$\delta, \beta, \theta > 0$
Xie and Lai [25]	$1 - \mathrm{e}^{-(ax)^b - (cx)^d}$	$a, b, c, d > 0$

We again use the maximum likelihood method to estimate the parameters of each distribution. For computational reasons we use Mathematica instead of R to maximize the likelihood. We use the functions `FindDistributionParameters` (for implemented distributions), where we increase the maximum number of iterations to a thousand, and `NMaximize` (for not implemented distributions), where we manually implement the log-likelihood.

To evaluate the models, we compare the models with the *Kaplan-Meier estimator*, which is a non-parametric statistic. The Kaplan-Meier estimator estimates the survival function, also called reliability function, of right-censored data. The survival function S of a random variable X at point t is defined as the probability that X is larger than t, i.e. $S(t) = \mathbb{P}(X > t) = 1 - F(t)$. In [26] Kaplan and Meier introduce the statistic $P(t)$, which can be interpreted as the proportion of the population that survives beyond time t.

In Figure 3.3 we plot the Kaplan-Meier estimator and the fitted models. The Kaplan-Meier estimator is calculated in R with the package `survival` [27]. In the figure we see that the Weibull, Hjorth and Lai et al. distribution match the non-parametric estimator better for large values of the working hours. The low values of the reliability here explain the lower expected values of these distributions in Table 3.6.

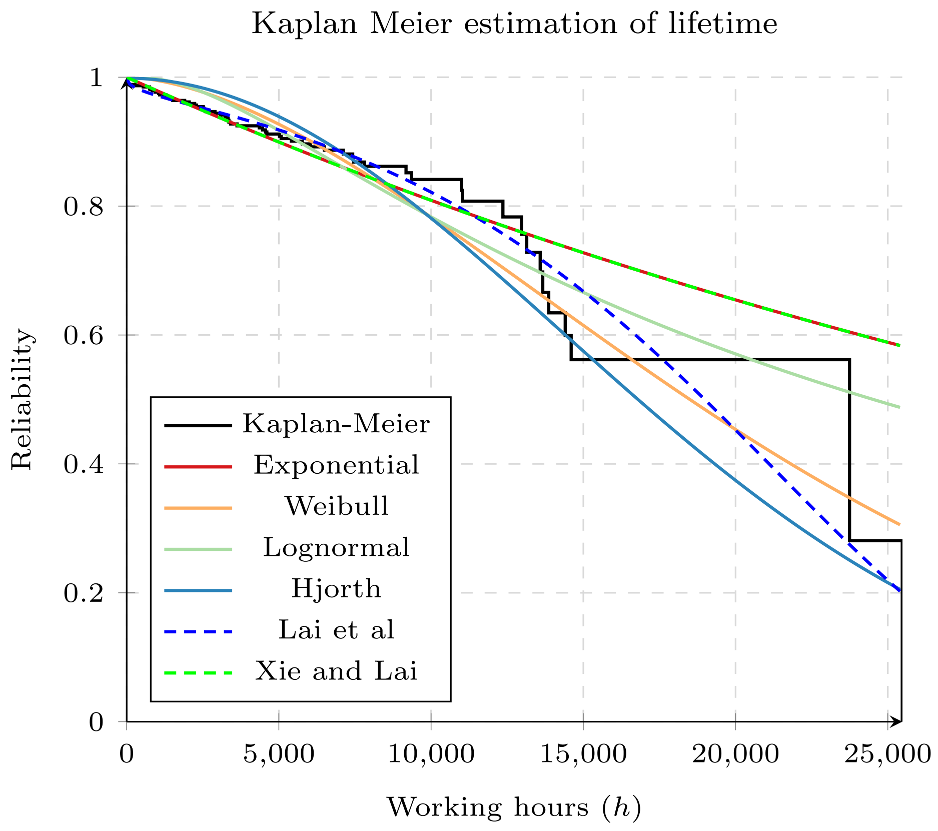

Figure 3.3: The Kaplan-Meier estimator and the fitted parametric models for the working hours at failure.

To further quantify the fit, we calculate the *mean squared error* (MSE) of the survival probabilities of the Kaplan-Meier estimator and the theoretical values of the fitted models. An estimate for the MSE for distribution F with observations $x_1, x_2, \ldots, x_n$ is given by

$$MSE(F) = \frac{1}{n}\sum_{i=1}^{n}(KM(x_i) - (1 - F(x_i)))^2,$$

where $KM(x_i)$ is the Kaplan-Meier estimate of the survival function at time x_i.

We choose the Weibull distribution to model the working hours at failure, since the estimated parameters yield a reasonable mean and the MSE is low. Furthermore, the Weibull distribution is well understood, commonly used in modeling component lifetime and available in most statistical software. The Lai et al. distribution also has a reasonable mean and a lower MSE and AIC than the Weibull distribution, but the Lai et al. distribution does not have the additional advantages of the Weibull, which is the reason why we do not choose the Lai et al. distribution.

Table 3.6: The censored fits of the working hours at failure of the big screen monitors

Distribution	**Parameter**	**Estimate**	**Mean**	**AIC**	**MSE**
Exponential	λ	2.119×10^{-5}	4.719×10^{4}	1.437×10^{3}	1.733×10^{-3}
Weibull	k	1.692	2.050×10^{4}	1.492×10^{3}	1.264×10^{-3}
	λ	2.297×10^{4}			
Lognormal	μ	1.011×10^{1}	4.724×10^{4}	1.683×10^{3}	1.557×10^{-3}
	σ	1.145			
Hjorth	β	2.162×10^{4}	1.787×10^{4}	1.453×10^{3}	1.713×10^{-3}
	δ	4.902×10^{-9}			
	θ	1.942			
Lai et al.	λ	1.716×10^{-3}	1.814×10^{4}	1.413×10^{3}	5.652×10^{-4}
	β	3.928×10^{-1}			
	ν	1.123×10^{-4}			
Xie and Lai	a	2.120×10^{-5}	4.718×10^{4}	1.443×10^{3}	1.733×10^{-3}
	b	1.000			
	c	0.			
	d	2.750×10^{-3}			

Degradation For the degradation values at failure, we must choose a distribution with a bounded support, such as the beta distribution. We use Equation (3.1) to transform the brightness values in the training set to degradation values and scale the degradation values to the interval [0,1]. Using the scaled values we obtain, in a same way as for the working hours, the estimates $\alpha = 1.216$ and $\beta = 0.7149$ for the beta distribution.

The expected value of the degradation at failure, which is 220, seems low and the expected value of the working hour is high for many distributions. We discuss one of the reasons in Section 10.2, where the ratio between censored data and uncensored data in the training set is treated.

Environment There are too few measurements in the training set of feature 5 to get reliable parameter estimates. We therefore choose to model the time until the environment changes state as a random variable following a Weibull distribution, just as the working hours. We assume that a monitor fails, when feature 5 is larger than 0. We therefore choose the parameters for the Weibull distribution such that $\mathbb{P}(E < L)$ is similar to the fraction of instances in the training set for which feature 5 is larger than 0. Parameters that satisfy the condition are $\lambda = 50,000$ and $k = 2.8$, which are used in the rest of the report.

Chapter 4

Policies Under Consideration

In this chapter we discuss some policies that prescribe when a monitor should be replaced to minimize the objective of Equation (1.1). The idea of a policy is to replace the monitors before a failure occurs. In Chapter 3 we make the assumption that a failure of a monitor is caused by one of three reasons; either the working hours, the age or the environment causes the failure. A feasible solution to the optimization problem is a set of rules, i.e. a policy, that specifies for all possible values of the three quantities whether a replacement should be made or not.

However, the set of policies is infinite. To see this, we only need to consider the plane spanned by the working hours and degradation. Each policy can be seen as a curve that divides the positive quadrant of the plane spanned into two: one part where no preventive maintenance actions are planned and one where preventive replacements are prescribed. The curve divides the plane into two and a policy is thus characterized by a curve. Since there are an infinite number of curves, the set of policies is infinite.

Luckily, many feasible solutions can be discarded, since they do not specify a policy that is interpretable or have undesirable properties. For instance, a curve that can be crossed several times by an increasing jump process is, with our assumptions, undesirable. In such as policy, no one-to-one relationship exists between the location on the plane and whether a replacement has been scheduled or not. The question that remains is thus which solutions are acceptable and how they can be ordered to decrease the complexity of the optimization problem.

We answer that question in this chapter. Instead of removing all the undesirable policies from the solution space, we discuss three classes that have desirable properties. In the remainder of the report our focus is on these three classes.

All the policies we look at in this report are stationary policies, but we do not have a guarantee that the global optimal policy is stationary. However, we do know that for Markov decision processes, with deterministic static costs and two possible actions per state (perform maintenance or do nothing), the optimal action is a state dependent stationary deterministic policy. Even though our model is not an MDP, it seems reasonable that the stationary policies at least indicate the nature of the global optimal policy.

Before we discuss the three classes, we recall one assumption made in Chapter 3, namely that there is enough time to schedule and perform a replacement after the environment switches from 0 to 1. A desired property is thus that a policy specifies a replacement after the environment switches. All policies that we discuss have this property and the differences between the policies are thus the curves in the plane spanned by the working hours and degradation. Because of this, we specify the curve when talking about a policy in the remainder of the report.

The three classes that we discuss next are related to the modelling of the failures. Each class is based on an assumption of the failure. The first class assumes that a failure occurs after some univariate thresholds are crossed; the second class assumes that a failure occurs when the probability of the monitor working crosses some fixed threshold; the third class assumes that there is some linear dependence between the working hours and degradation which describes the failure.

4.1 Class 1: Univariate Thresholds

The first class of policies is described by two thresholds: one for the working hours and one for the degradation. We denote the thresholds by τ_{WH} and τ_D for the thresholds on the working hours and degradation, respectively. Together the thresholds specify a rectangle in the plane, see the solid blue curve in Figure 4.1. A replacement is planned whenever the degradation process crosses either threshold. In Figure 4.1 a repair is thus made when the degradation process (solid black line) crosses the threshold for the working hours.

One situation in which the class 1 policies are a natural choice is a scenario where the degradation and working hours are completely independent. Then the information one has on either of them does not give any information on the other. Therefore, for both the degradation and working hours a logical policy is a threshold policy.

An advantage of the policies in class 1 is that the reasons to replace a component are clear. Even though in most settings there is no independence, the clear rules of the policies make the policies interesting.

4.2 Class 2: Ellipsoids

The second class consists of ellipse shaped curves that represent the points in the plane at which the probability of failure is η for some η between 0 and 1. For a given point in time t where the degradation is d, the probability that a monitor has not failed is the probability that neither of the three causes of failure have occurred. The probability of being alive at point (t, d, e) is thus given by

$$\begin{aligned}\mathbb{P}(\{L > t\} \cap \{E > t\} \cap \{D > D(t)\} \mid D(t) = d, Env(t) = e)&\\ = \mathbb{P}(L > t)\,\mathbb{P}(D > d)\,\mathbb{1}\{e = 0\}\,, &\quad (4.1)\end{aligned}$$

where, we can split the left-hand side due to the assumed independence. When we plot the solutions of equalizing Equation (4.1) to a constant η, we obtain a curve in the plane that looks like an ellipse, see the dotted red curve in Figure 4.1. A value of η close to 1 results in a curve close to the origin, while a value close to 0 results in a curve further away.

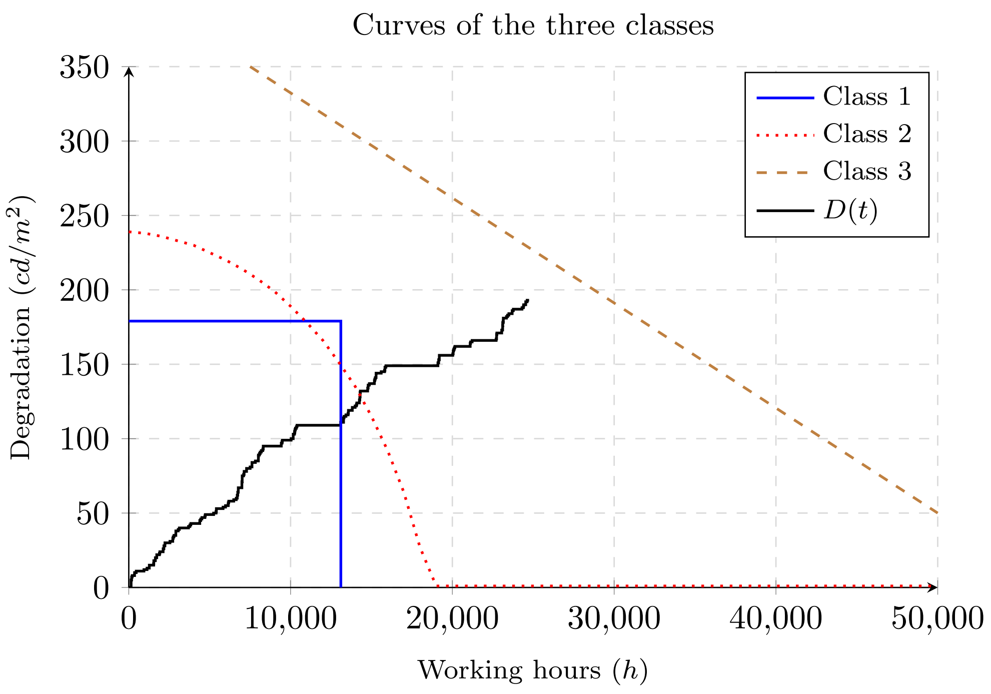

Figure 4.1: Possible curves of the three policy classes and a possible degradation process from start to failure. All policies specify a replacement when the associated curve is crossed by the degradation process.

One appealing property of the second class is that we have an easy approximation of the fraction of monitors that is replaced preventively, namely η. The reason that it is only an approximation is that the replacements caused by a change in the environment are not accounted for. Furthermore, we know what the probability is that a monitor is still alive when we want to replace it, which is helpful for the interpretability.

4.3 Class 3: Linear Planes

The third class of policies consists of linear planes, which include the classifiers obtained by the SVM algorithm. Linear planes are given by the solutions (t, d) of the equation

$$t + a \cdot d = a \cdot b \tag{4.2}$$

for some real constants a and b. The constant b corresponds to the intercept of the plane with the degradation axis in Figure 4.1. The solutions of Equation (4.2) lie on a line, see the brown dashed line in Figure 4.1. In the figure, we see that the monitor failed before the linear plane was crossed, so, in this case, corrective maintenance was performed.

One advantage of the linear planes is that they can take some dependence between the working hours and degradation into account. Often one of the two is the dominant factor in the policy. For instance, a plane that start at the same point as threshold for the degradation of class 1 in Figure 4.1 and then slowly decreases, sees the degradation as the dominant factor, while acknowledging that the working hours also have some contribution to the failure occurrences. Another advantage is that the classifiers found by the SVM algorithm are part of the third class, thus the method of the report includes the old classifiers.

Two desired properties that all three classes have is that the curves can be described as smooth and continuous. There are no ‘holes’ in the curves that allow sample paths of the degradation process to never cross the curve. Neither can there be sample paths that cross a curve multiply times, which would be undesirable with our assumptions on the stochastic quantities. Under other assumptions, these properties could be undesired, but, since this is not the case in the project, we do not look closer into this.

Chapter 5

Average Cost Criterion

In this chapter we discuss the first performance indicator, namely the operational costs. We define the operational costs for a policy as the costs that are made when following that policy. We assume that the total cost is made up of two parts: the cost for a corrective replacements and the cost of a preventive replacement. The two costs are assumed to cover all relevant costs to a predictive or corrective replacement, e.g. shipping costs and loss of income due to the downtime of the system.

The operational costs for the current policy, which is based on an SVM classifier, is presumably relatively high, since no costs were taking into account during the training of the SVM classifier. Therefore, the related policy does not take the costs into account. In the literature one can find methods to include the cost in a machine learning classifier, see, for instance, [28] and [29], but these methods do not work well for our problem, which we discuss in more detail in Chapter 9. As discussed in Chapter 1, policies obtained by Markov decision process (MDP) or similar methods do include costs and the performance indicator we propose is based on literature in this area.

In the following pages, we derive the general form of the performance indicator for the operational costs and discuss the influence of the three classes of Chapter 4 on the general form.

5.1 The Average Cost

When one follows a maintenance policy, one of two events happen to a monitor: the monitor fails during a period where the policy prescribes that no action should be taken, or the policy prescribes that a monitor should be replaced preventively. In the first scenario one must perform corrective maintenance with associated *corrective maintenance cost* c_c. In the second scenario one performs preventive maintenance and pays the *preventive maintenance cost* c_p. After the monitor is replaced, the same two events can happen. We call the time between two consecutive replacements a *cycle*.

The operational costs are a function of the costs made in each cycle. Two distinctions are made in the literature on MDPs when looking at the costs (see [30]); firstly, the replacement costs can be discounted or undiscounted; secondly, the considered time horizon can be finite or infinite. We choose to consider undiscounted costs, i.e. we value the amounts c_c and c_p the same now as we do in several years, and look at an infinite time horizon.

The reason for this combination is that we consider many monitors that should not be replaced too often before newer versions, with different characteristics, replace them. Since there are many monitors, the limiting results should give reasonable results. Furthermore, since there are few replacements per monitor in the timespan between versions, the discounting plays less of a role.

The lifetimes of all the monitors under Philips' care can thus be partitioned into cycles. The total cost for Philips in all cycles is one possible performance indicator. However, as time progresses, the total cost would go to infinity. Fixing the number of cycles prevents this, but then replacing all monitors immediately gives the lowest cost. Dividing the total costs in all cycles by the total length of all the cycles, effectively measuring the costs made per time unit, has neither of these problems and is called the *average cost criterion.*

We assume that the total cost of a corrective and preventive replacement are c_c and c_p respectively. We further assume that c_c is larger than c_p, i.e. $c_c > c_p$, and that both are positive reals. If $c_c \leq c_p$, then a corrective maintenance policy is always the optimal option.

Let T denote the time at which a monitor fails (see Equation (3.4)) and S the time when a policy prescribes a preventive replacement. The cycle length is then given by the minimum of T and S. Because of our assumptions, the cycle lengths of all monitors are i.i.d. and so are the costs made at the end of a cycle, which we denote by R. We can thus consider all cycles as belonging to one monitor, which produces one long sequence of cycles and costs. This monitor can be described by a renewal reward process, where the cycle lengths are the interarrival times and the costs are the jump sizes. For more information on renewal reward processes see [31], especially Section 7.4.

The total costs until time t, denoted by $V(t)$, is given by the sum of the costs of the cycles finished by time t, so

$$V(t) = \sum_{i=1}^{M(t)} R_i,$$

where $M(t)$ is the number of finished cycles until time t, defined similarly as $N(t)$ in Equation (3.3), and R_i the costs in cycle i. To obtain the average cost, we divide the total cost $V(t)$ by t and let the limit go infinity. The average cost $\widetilde{g}$ is thus defined as

$$\widetilde{g} = \lim_{t\to\infty} \frac{V(t)}{t}. \tag{5.1}$$

Theorem 5.1 gives the limiting result of Equation (5.1), which simplifies the expression of the average cost. Generally, the theorem is defined for rewards instead of costs, but for consistency we formulate the theorem in terms of costs. We refer to [32] (Theorem 3.6.1) for a proof of Theorem 5.1.

Theorem 5.1 (The elementary renewal reward theorem, Theorem 3.6.1 in [32]). *Let $\{M(t), t \geq 0\}$ be a renewal process with positive interarrival times $Z_1, Z_2, \ldots$ and let $R_1, R_2, \ldots$ be the costs at the arrival times. Assume that $\mathbb{E}[Z] < \infty$ and $\mathbb{E}[|R|] < \infty$. Furthermore, we assume that $(Z_i, R_i)_{i \geq 1}$ is a sequence of i.i.d random variables. Let $V(t) = \sum_{i=1}^{M(t)} R_i$ be the total cost up to time t. It holds that*

$$\widetilde{g} = \lim_{t \to \infty} \frac{V(t)}{t} = \frac{\mathbb{E}[R]}{\mathbb{E}[Z]} = \frac{\textit{Expected cost per cycle}}{\textit{Expected length of cycle}}. \tag{5.2}$$

Denoting the average cost for a policy $\boldsymbol{\sigma}$ by $\widetilde{g}(\boldsymbol{\sigma})$ and remembering the fact that there are only two possible costs per cycle, we can write the average cost of policy $\boldsymbol{\sigma}$ as

$$\widetilde{g}(\boldsymbol{\sigma}) = \frac{c_c \mathbb{P}(T \leq S_{\boldsymbol{\sigma}}) + c_p \mathbb{P}(T > S_{\boldsymbol{\sigma}})}{\mathbb{E}[\min\{T, S_{\boldsymbol{\sigma}}\}]}, \tag{5.3}$$

where the subscript in $S_{\boldsymbol{\sigma}}$ indicates that the quantity depends on policy $\boldsymbol{\sigma}$.

From Equation (5.3) we deduce that only the relative value of the corrective and preventive maintenance costs play a role for the location of the minimum and not the actual values. We can therefore choose c_p equal to 1. The value of c_c then corresponds to the ratio between the two, or, in other words, how much more expensive a corrective replacement is compared to a preventive replacement.

A consequence of fixing c_p is that the average cost will be close to 0, since we assume that the expected cycle length is much larger than the ratio of the costs. To make it easier to compare the average cost term with the performance term, which we discuss in Chapter 6, we multiply the average cost with the expected lifetime of the monitor and get

$$g(\boldsymbol{\sigma}) = \frac{c_c \mathbb{P}(T \leq S_{\boldsymbol{\sigma}}) + c_p \mathbb{P}(T > S_{\boldsymbol{\sigma}})}{\mathbb{E}[\min\{T, S_{\boldsymbol{\sigma}}\}]} \cdot \mathbb{E}[T] , \tag{5.4}$$

which is the performance indicator that we use in the report and the first term of Equation (1.1). Note that the definition of $g(\boldsymbol{\sigma})$ ensures that $g(\boldsymbol{\sigma})$ is larger than 1, since $\mathbb{E}[\min\{T, S_{\boldsymbol{\sigma}}\}] \leq \mathbb{E}[T]$.

5.2 Influence of Classes on Cost Criterion

To evaluate the expression in Equation (5.4) for some $\boldsymbol{\sigma}$ we need the distributions of T and $S_{\boldsymbol{\sigma}}$. The first is independent of the three policy classes, which we discuss in Chapter 4, the latter is not. However, the probability functions of both T and $S_{\boldsymbol{\sigma}}$ need an expression for the number of jumps at time t, given by $N(t)$. We assumed that the interarrival times are lognormal distributed and the number of jumps at time t is therefore given as an convolution product without a closed form. In Chapter 7 we briefly discuss the computational problems when numerical approximating $N(t)$ and the probability density functions of T and $S_{\boldsymbol{\sigma}}$.

If we would assume exponentially distributed interarrival times, then we would have a compound Poisson process. We would then have a closed expression for $N(t)$, but other problems would still remain. For instance, the problem of finding the hitting probability of a compound Poisson process with a linear boundary is well studied and known to be a hard problem. For the compound Poisson process, some results are summarized in [33]. The results in this paper rely on the exponential assumptions in the Poisson process and are therefore not easily extended. Other research, for instance [34], has focused on finding Laplace transforms of the first passage time density, but these results are also not directly applicable.

Chapter 6

Performance Component

In the previous chapter we discuss the first performance indicator, namely the average cost criterion. In this chapter we discuss how we can incorporate a second performance measure in the decision-making process.

In this chapter we first look at the question of why we need to incorporate a second performance measure and which measure is suitable. We then look at two different ways to incorporate the performance measure and the theoretical formulation of the measure under the three policy classes of Chapter 4.

6.1 The Precision Measure

Apart from saving costs, the moment of replacement is also important in the decision-making process. The current situation at Philips is that Philips prioritizes the avoidance of erroneous classifications that result in too early replacements, i.e. a replacement when the component would still have functioned correctly for a long time, over missed opportunities to replace a component preventively. Philips' standpoint is influenced by the wants of their customers, who are often not (yet) willing to replace components if the old ones are still functioning satisfactory.

Policies only considering operational costs, e.g. minimizers of Equation (5.4), may not comply to the request of avoiding too early replacements. Therefore, to comply to the request, we need a way to measure the compliance of a policy to the request. The measure can then be added to the average costs to find a trade-off.

To select a measure, we draw inspiration from the current SVM based model and, more generally, from the machine learning field. In machine learning the performance of a classifier is determined by a function of the number of *true positives* (`TP`), *true negatives* (`FN`), *false positives* (`FP`) and *false negatives* (`FN`) on a set. Since the overwhelming majority of measurements are of machines that are working and which are classified as working, we do not look at all available data points when calculating the performance measure. Instead, we look at the moment when a component is replaced. We define true positives as preventive replacements that are performed 30 days before the component would have failed and false positives as preventive replacements that are performed more than 30 days before failure. The 30 days are specified by Philips in the current SVM model and therefore adopted here. The interval of 30 days before a failure is called the *predictive interval.*

The objective of Philips can now be formulated as minimizing the number of false positives. In our setting, the objective can be reformulated as maximizing the precision measure. The precision is defined as (see, for instance, [11])

$$precision = \frac{\texttt{TP}}{\texttt{TP} + \texttt{FP}} = \frac{\text{Number of correctly predicted failures}}{\text{Total number of predicted failures}}. \tag{6.1}$$

6.2 Penalty Terms

There are many ways in which the precision can be included in the decision-making process. In this project we propose to add a penalty term to the average cost of Equation (5.4), which is based on the precision and a target value of the precision. In this section we discuss two penalty terms and discuss their interpretation, how they differ and, where possible, their use in other areas in the literature.

Before we introduce the penalty terms, we note that the precision is per definition between 0 and 1, and that the average cost, defined in Equation (5.4), is always larger than 1. To balance the two and be able to put more or less weight on the precision, we introduce a real-valued, positive constant k. A high value of k means that we put a lot of weight on the precision, which is done if we are more interested in fully utilizing the lifetime of a monitor than cutting down on costs.

Let α be a target value of the precision, so α lies between 0 and 1, and let $prec(\boldsymbol{\sigma})$ be the precision obtained by policy $\boldsymbol{\sigma}$. We denote a penalty function by ℓ, which takes as input a policy and a target value and returns a real number, so $\ell : \mathcal{S} \times [0,1] \to \mathbb{R}$, where $\mathcal{S}$ is the set of all policies.

The first loss function that we consider is an exponential loss function. We define the *exponential loss* as

$$\ell(\boldsymbol{\sigma}, \alpha) = k\left(\mathrm{e}^{\alpha - prec(\boldsymbol{\sigma})} - 1\right). \tag{6.2}$$

The -1 in Equation (6.2) ensures that policies with a precision higher than the target are rewarded, while the others are penalized. The reward increases as the precision increases and therefore optimal policies will have high precision values, if the weight is large enough. One algorithm that uses an exponential loss function is the AdaBoost algorithm.

The increasing reward for higher precision values of the exponential loss may not always be wanted. A loss function that only penalizes policies with a precision lower than the target is the *hinge loss.* A penalty based on the hinge loss is

$$\ell(\boldsymbol{\sigma}, \alpha) = k \max\{\alpha - prec(\boldsymbol{\sigma}), 0\}. \tag{6.3}$$

The hinge loss penalizes policies that have a precision lower than the target value, but does not differentiate between policies with a higher precision. Since a higher precision often implies a higher cost, using the hinge loss will result in an optimal policy that has a precision close to the target, if k is large enough. The hinge loss is an common loss function in machine learning, which is, for instance, used in SVMs.

For both loss functions it holds that the value of a “large enough value of k” depends on the corrective maintenance costs c_c. In Chapter 7 we discuss that there is a switching point where the dominant term changes from the costs to the precision.

6.3 Influence of Classes on Precision

Assuming that we have the distributions of T and $S_{\boldsymbol{\sigma}}$, then we can express the theoretical value of the precision of a policy. Using the distributions of T and $S_{\boldsymbol{\sigma}}$ to express the quantities in Equation (6.1), we get that

$$prec(\boldsymbol{\sigma}) = \frac{\mathbb{P}(S_{\boldsymbol{\sigma}} \leq T \leq S_{\boldsymbol{\sigma}} + z)}{\mathbb{P}(S_{\boldsymbol{\sigma}} \leq T)} = \mathbb{P}(S_{\boldsymbol{\sigma}} \leq T \leq S_{\boldsymbol{\sigma}} + z \mid S_{\boldsymbol{\sigma}} \leq T)\,.$$

The numerator represents the replacements inside the predictive interval and thus the correctly predicted failures. The denominator represents all monitor replacements.

Chapter 7

Numerical Approximations to Solution

In the previous chapters we construct the two terms of the main optimization problem. In the current chapter we combine the results of Chapter 5 and Chapter 6 to obtain the optimization problem as formulated in the introduction (see Equation (1.1) in Chapter 1) and discuss some numerical complications of solving the optimization problem. We end the chapter with methods to approximate the solutions, which are used in the next chapter to compare different policies.

Filling in the average cost criterion of Equation (5.4) and the exponential and hinge loss functions stated in Equations (6.2) and (6.3) in the main optimization problem stated in Equation (1.1), we get

$$\underset{\boldsymbol{\sigma}\in\mathcal{S}}{\arg\min}\, \frac{c_c\mathbb{P}(T \leq S_{\boldsymbol{\sigma}}) + c_p\mathbb{P}(T > S_{\boldsymbol{\sigma}})}{\mathbb{E}[\min\{T, S_{\boldsymbol{\sigma}}\}]} \cdot \mathbb{E}[T] + k\max\{\alpha - prec(\boldsymbol{\sigma}), 0\}, \tag{7.1}$$

and

$$\underset{\boldsymbol{\sigma}\in\mathcal{S}}{\arg\min}\, \frac{c_c\mathbb{P}(T \leq S_{\boldsymbol{\sigma}}) + c_p\mathbb{P}(T > S_{\boldsymbol{\sigma}})}{\mathbb{E}[\min\{T, S_{\boldsymbol{\sigma}}\}]} \cdot \mathbb{E}[T] + k\left(\mathrm{e}^{\alpha - prec(\boldsymbol{\sigma})} - 1\right). \tag{7.2}$$

In the rest of the chapter we discuss complications with solving the problems in Equations (7.1) and (7.2) and how we approximate the solutions.

7.1 Numerical Complications

In theory, the problems in Equations (7.1) and (7.2) are fully known and can be solved by some optimization method, although a unique optimal solution is not guaranteed. However, as we discuss in Section 5.2, we lack the explicit expression of the number of jumps at time t, i.e. $N(t)$.

One solution is to use Monte Carlo simulation to approximate the distribution of $N(t)$. The found approximation can then be used to approximate the other cumulative distribution functions. For the probability functions, one can use a finite difference formula, e.g. the symmetric difference quotient. However, it turns out that the computational power to reach an acceptable accuracy is too high for practical use.

7.2 Numerical Approximations

Instead of approximating the distribution of $N(t)$, we use approximations to the whole objective function. For each class we use a different technique, which we discuss in this section.

7.2.1 Class 1

In Section 4.1 we discuss the first class of policies, which consists of two thresholds. The assumption here is that the working hours and degradation at failure are independent and therefore a threshold for each of them is logical.

To approximate the thresholds, we decompose the process. For the threshold on the working hours, we assume that failures are only caused by the aging of the monitor. Similarly, for the degradation thresholds we assume that only the degradation causes failures. With these assumptions we can calculate the two thresholds using the optimality theorem stated in Theorem 7.1.

Before we give the optimality result in Theorem 7.1, we need to introduce the notion of the hazard rate of a component. The *hazard rate* can be interpreted as the probability that the component fails within the next infinitesimal time interval, given that it has survived so far. Formally, the hazard rate is defined as (see [32]):

$$h(t) = \lim_{h \to 0} \frac{\mathbb{P}(T \leq t + h \mid T \geq t)}{h} = \frac{f(t)}{1 - F(t)} = -\frac{\mathrm{d}}{\mathrm{d}x} \log(1 - F(t))\,.$$

Theorem 7.1. *Assume that the lifetime of a component is a positive random variable T with distribution function $F_T(t)$. The average cost, see Equation (5.3), for a threshold $\boldsymbol{\sigma}$, which is a positive real, is given by*

$$\widetilde{g}(\boldsymbol{\sigma}) = \frac{c_c \mathbb{P}(T \leq \boldsymbol{\sigma}) + c_p \mathbb{P}(T \geq \boldsymbol{\sigma})}{\int_0^{\boldsymbol{\sigma}} 1 - F_T(t)\,\mathrm{d}t}. \tag{7.3}$$

The minimum of this function is attained at a finite $\boldsymbol{\sigma}$ if and only if the hazard rate has a increasing tail. In this case it is worthwhile to do preventive maintenance. If the hazard rate has a decreasing tail the minimum is never attained and the optimal policy is the corrective one.

For the proof of Theorem 7.1, we refer to Appendix C. The intuition behind the result is that if the hazard rate is decreasing a component becomes more reliable over time and does not need to be changed. However, if the hazard rate increases, the probability of the component failing keeps increasing and it is profitable to change the component preventively.

Threshold Working Hours For the working hours, we directly apply the theorem using the working hours at failure, i.e. L, as the controlling distribution.

For the precision, we have that

$$\begin{aligned} prec(\boldsymbol{\sigma}) &= \frac{\text{Number of correctly predicted failures}}{\text{Total number of predicted failures}} \\ &= \frac{\mathbb{P}(\boldsymbol{\sigma} \leq L \leq \boldsymbol{\sigma} + z)}{\mathbb{P}(L \geq \boldsymbol{\sigma})} \\ &= \frac{\mathbb{P}(L \geq \boldsymbol{\sigma}) - \mathbb{P}(L \geq \boldsymbol{\sigma} + z)}{\mathbb{P}(L \geq \boldsymbol{\sigma})} \\ &= 1 - \frac{\mathbb{P}(L \geq \boldsymbol{\sigma} + z)}{\mathbb{P}(L \geq \boldsymbol{\sigma})}, \end{aligned} \tag{7.4}$$

where z is the predictive interval of 30 days in which we call a replacement correct. From the data, we estimate that the average number of working hours per day is 10, so we use $z = 30 \cdot 10 = 300$.

Filling in Equation (7.3) and Equation (7.4) in the main optimization problem gives us an expression which we optimize using the `optimise()` function in R. We use the same function to obtain the thresholds for the degradation.

In Table 7.1, the results for various cost ratios are shown. The table shows that there is no solution for the working hours threshold for a cost ratio of 2 and $k = 2$. The reason for this is that the precision dominates the objective and keeps increasing as τ_{WH} increases. The objective therefore keeps decreasing. For larger cost ratios, the average cost starts to dominate and we obtain a unique solution. We notice that both thresholds decrease as the ratio increases. When we use smaller thresholds, we replace the monitor sooner. This behavior is the same for all classes.

Threshold Degradation The univariate threshold for the degradation is a bit more complex. The problem with the degradation is that the decision variable, i.e. the degradation, does not contain any information on the lifetime of a monitor, which means that we can not calculate the average cost. However, if we use the degradation process, which includes the time, we can calculate everything.

To calculate the expected cycle length and costs, we first define the following stopping times

$$M_{\boldsymbol{\sigma}} = \min\left\{ m \in \mathbb{N} \mid \sum_{i=1}^{m} Y_i \geq \boldsymbol{\sigma} \right\}, \; M_D = \min\left\{ m \in \mathbb{N} \mid \sum_{i=1}^{m} Y_i \geq D \right\}$$

$$M = \min\{M_{\boldsymbol{\sigma}}, M_D\} = \min\left\{ m \in \mathbb{N} \mid \sum_{i=1}^{m} Y_i \geq \min\{\boldsymbol{\sigma}, D\} \right\}$$

The expected cost is then given by

$$\mathbb{E}[costs] = c_p \mathbb{P}(M_{\boldsymbol{\sigma}} < M_D) + c_c \mathbb{P}(M_{\boldsymbol{\sigma}} \geq M_D)\,. \tag{7.5}$$

The probability $\mathbb{P}(M_{\boldsymbol{\sigma}} < M_D)$ can be written as

$$\begin{aligned}
\mathbb{P}(M_{\boldsymbol{\sigma}} < M_D) &= \mathbb{P}\left(\sum_{i=1}^{M_{\boldsymbol{\sigma}}} Y_i < D\right) \\
&= \sum_{k=1}^{BL} \mathbb{P}\left(\sum_{i=1}^{k} Y_i < D\right) \mathbb{P}(M_{\boldsymbol{\sigma}} = k) \\
&= \sum_{k=1}^{BL} \left[\sum_{d=1}^{BL} \mathbb{P}(d < D)\, \mathbb{P}\left(\sum_{i=1}^{k} Y_i = d\right)\right] \mathbb{P}(M_{\boldsymbol{\sigma}} = k) \\
&= \sum_{k=1}^{BL} \left[\sum_{d=k}^{BL} \mathbb{P}(d < D) \binom{d-1}{k-1} p^k (1-p)^{d-k}\right] \mathbb{P}(M_{\boldsymbol{\sigma}} = k)\,.
\end{aligned}$$

In the last equality we use the fact that a sum of geometric random variables follows a negative binomial distribution. We use the same fact for the probability $\mathbb{P}(M_{\boldsymbol{\sigma}} = k)$, which equals

$$\mathbb{P}(M_{\boldsymbol{\sigma}} = k) = \mathbb{P}\left(\sum_{i=1}^{k-1} Y_i < \boldsymbol{\sigma}, \sum_{i=1}^{k} Y_i \geq \boldsymbol{\sigma}\right) = \sum_{d=1}^{\boldsymbol{\sigma}-1} \mathbb{P}\left(\sum_{i=1}^{k-1} Y_i = d\right) \mathbb{P}(Y_k \geq \boldsymbol{\sigma} - d)\,.$$

We can then compute Equation (7.5) by using the fact that $\mathbb{P}(M_{\boldsymbol{\sigma}} \geq M_D) = 1 - \mathbb{P}(M_{\boldsymbol{\sigma}} < M_D)$.

To calculate the expected length, we first look at the probability that M is larger than n for n between 0 and $\boldsymbol{\sigma} - 1$. Using the definitions of $M_{\boldsymbol{\sigma}}$ and M_D, we obtain the following

$$M > n \;\Leftrightarrow\; M_{\boldsymbol{\sigma}} > n \;\wedge\; M_D > n \;\Leftrightarrow\; \sum_{i=1}^{n} Y_i < \boldsymbol{\sigma} \;\wedge\; \sum_{i=1}^{n} Y_i < D \tag{7.6}$$

for $n = 0, 1, \ldots, \boldsymbol{\sigma} - 1$. Using the relations in Equation (7.6), we get that

$$\mathbb{P}(M > n) = \mathbb{P}\left(\sum_{i=1}^{n} Y_i < \boldsymbol{\sigma}, \sum_{i=1}^{n} Y_i < D\right) = \sum_{k=n}^{\boldsymbol{\sigma}-1} \mathbb{P}\left(\sum_{i=1}^{n} Y_i = k\right) \mathbb{P}(D > k)\,.$$

Using the facts that for any non-negative random variable Z the expectation is given by $\mathbb{E}[Z] = \sum_{z=0}^{\infty} \mathbb{P}(Z > z)$ and that $\mathbb{E}\left[\sum_{i=1}^{N} X_i\right] = \mathbb{E}[N]\,\mathbb{E}[Y]$ for a nonnegative integer-value random variable N and a sequence of i.i.d. random variables X that are independent of M, we get that

$$\mathbb{E}[\mathit{length}] = \mathbb{E}\left[\sum_{i=1}^{M} X_i\right] = \mathbb{E}[X] \sum_{n=0}^{\boldsymbol{\sigma - 1}} \mathbb{P}(M > n)\,.$$

The precision is, following a similar reasoning as in Equation (7.4) and using the memorylessness of the geometric distribution, given by

$$\begin{aligned}
prec(\boldsymbol{\sigma}) &= \mathbb{P}\left(D \in \left[\boldsymbol{\sigma}, \boldsymbol{\sigma} + \tilde{Y} + \sum_{i=1}^{N(z)} Y_i\right] \mid D \geq \boldsymbol{\sigma}\right) \\
&= \frac{\mathbb{P}(D \geq \boldsymbol{\sigma}) - \mathbb{P}\Big(D \geq \boldsymbol{\sigma} + \tilde{Y} + \sum_{i=1}^{N(z)} Y_i\Big)}{\mathbb{P}(D \geq \boldsymbol{\sigma})} \\
&= 1 - \frac{\sum_{n=0}^{\infty} \mathbb{P}\Big(D \geq \boldsymbol{\sigma} + \tilde{Y} + \sum_{i=1}^{z} Y_i\Big)\, \mathbb{P}(N(z) = n)}{\mathbb{P}(D \geq \boldsymbol{\sigma})},
\end{aligned}$$

where $N(z)$ is the number of jumps in the predictive interval of length z. We see that we again need $\mathbb{P}(N(z) = n)$, but here an empirical distribution is satisfactory. We obtain the empirical distribution by simulating the arrival process and counting the number of jumps at time z, see Algorithm 7.1.

Algorithm 7.1: Procedure to obtain the empirical distribution of the number of jumps in the predictive interval.

```
Data: The number of simulated arrival process $N$, distribution of interarrival time and
      length of the predictive interval $z$.
Result: Empirical distribution of $N(z)$.
1 Initialize vector prob          // index starts at 0, $j$-th entry corresponds to
  $\mathbb{P}(N(z) = j)$
2 for $i$ in 1 to $N$ do
3     Draw $BL$ interarrival times $x_1, x_2, \dots, x_{BL}$
4     Calculate the arrival times $AT_j = \sum_{k=1}^{j} x_j$ for $j = 1, 2, \dots, BL$
5     Calculate number of arrivals, with arrivals $= \sum_{j=1}^{BL} \mathbb{1}\{AT_j \leq z\}$
6     increase prob[arrivals] by 1
7 Divide prob by $N$
```

7.2.2 Class 2

For the second and third policy classes we use similar approximation methods. We approximate the value of the objective function by simulating monitors and estimate the objective function for different policies.

To simulate a monitor, we use the assumption that the degradation process accurately describes the state of a monitor. Therefore, to simulate monitors we generate sample paths of the degradation process, see Algorithm 7.2.

The simulated paths are close to the paths of the monitor chosen to represent all monitors (see end Section 3.4.1). Since the parameters are fixed, there is less variation in the simulated paths than in the paths of the actual data. Furthermore, our definition of the moment of failure changes the distribution of the working hours at failure and the

Algorithm 7.2: The procedure to obtain one sample path. Note that path "continues" after it failed. One can also return a list with data up to and including the failure.

```
Data: Stochastic process description
Result: One sample path
1 Draw values at failure of the process (independently)
2 Draw $BL$ interarrival times and jump sizes (independently)
3 Calculate degradation process $D(t)$ (Equation (3.2))
4 Determine the moment of failure $T$ (Equation (3.4))
5 Label instances of $D(t)$ before $T$ as Normal and the rest as Failed
6 Return list of $t$, $D(t)$, $Env(t)$ and corresponding labels
```

degradation at failure. The reason for this is that the moment of failure is fixed when one of the three events happen. The results are that the other values are conditional on that event and that the values at failure are lower than what we expect from the distributions chosen in Section 3.4.2. One way to prevent this is to generate the paths is a way resembling the Brownian bridge. However, the first experiments in this direction did not give satisfactory results and due to time limitations we decided to follow the method we describe above instead of improving this alternative method.

To obtain an estimate for the value of the objective function, we simulate N paths, where N is large. For each sample path we determine the cost, age and whether a preventive replacement took place inside the predictive interval for a given policy, see Algorithm 7.3. The estimates of the average cost is then the total cost divided by the total age and the estimate for the precision is the number of correct replacements divided by the total number of replacement, see Line 13 and Line 14 in Algorithm 7.3.

The policies in the second class have one parameter, namely η, which is between 0 and 1. To find the optimal parameters we estimate the objective function for values of η between 0 and 1 with steps of 0.01. We use 10^4 simulated monitors.

In Figure 7.1 we see the estimates for a cost ratio between corrective and preventive maintenance of 10, $\alpha = 0.8$ and the hinge loss function. We notice two things: First, there is an unique minimum for both values of k. Second, the location of the minimum is dependent on the value of k. This observations confirms the discussions in Section 6.2, where we note that relative weight of the precision impact the optimal policy. For other values of k, the minimum will be either 0.89 or 0.03, which are the two minima in Figure 7.1. The results for other cost ratios are shown in Table 7.1.

Algorithm 7.3: Pseudo-code for the estimating the objective function of a policy. To estimate the value of the objective for more policies, an extra for loop around the if-statement is needed.

```
Input: A maintenance policy, number of sample paths ($N$), and values for $k$ and $\alpha$
Output: The value of he objective function for the maintenance policy
1  for n in 1 to N do
2      Generate one sample path according to Algorithm 7.2
3      $full.life[n] \leftarrow$ time of failure
4      if replacement did not occur before failure then
5          $age[n] \leftarrow$ time of failure
6          $cost[n] \leftarrow c_c$
7          $accurate[n] \leftarrow 0$
8      else
9          $age[n] \leftarrow$ time of replacement
10         $cost[n] \leftarrow c_p$
11         $accurate[n] \leftarrow \mathbb{1}\{$replacement in predictive interval$\}$
12 The expectation of $T$ is $\frac{1}{N}\sum_{i=1}^{n} full.life[i]$
13 The average cost estimate is $\frac{\sum_{i=1}^{N} cost[i]}{\sum_{i=1}^{N} age[i]}$
14 The precision estimate is $\frac{\sum_{i=1}^{N} accurate[i]}{\sum_{i=1}^{N} \mathbb{1}\{cost[i]=c_p\}}$
15 Calculate the objective function Equation (7.1) or Equation (7.2)
```

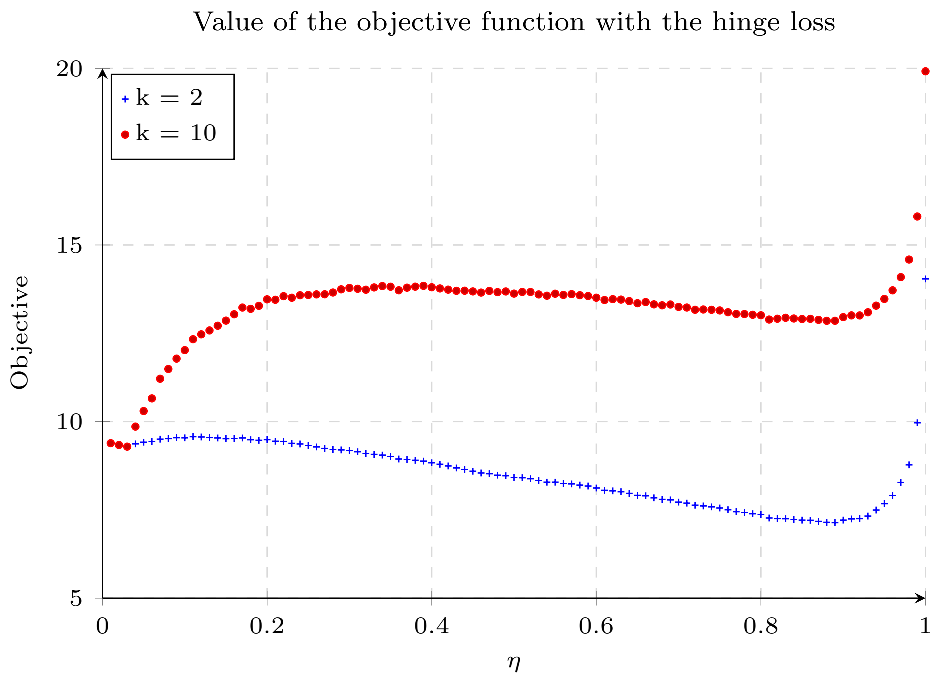

Figure 7.1: The objective function for class 2 policies for a cost ratio of 10, $\alpha = 0.8$ and two values of k.

7.2.3 Class 3

For the third class of policies we need to determine the slope a and intercept b of the hyperplane $t + a \cdot d = a \cdot b$. Similar to the second class, we approximate the objective function by simulating monitors. The policies we test have a slope between -150 and 2000 and an intercept between -300 and 600, both using step sizes of 50. We use 10^4 simulated monitors.

In Figure 7.2 we see the estimated values for a cost ratio of 10, $k = 2$ and $\alpha = 0.8$. When we compare the figure with Figure 7.1, we see that the minimum value is higher and the location of the minimum is less clear. For all cost ratios the objective function is lowest for policies with a positive slope and intercept, like the hyperplane in Figure 4.1. The results for other cost ratios are shown in Table 7.1.

The influence of k on the third class is similar to the influence of k on the second class, i.e. a higher k gives a more conservative policy. A more conservative policy in the third class has higher slope and intercept values.

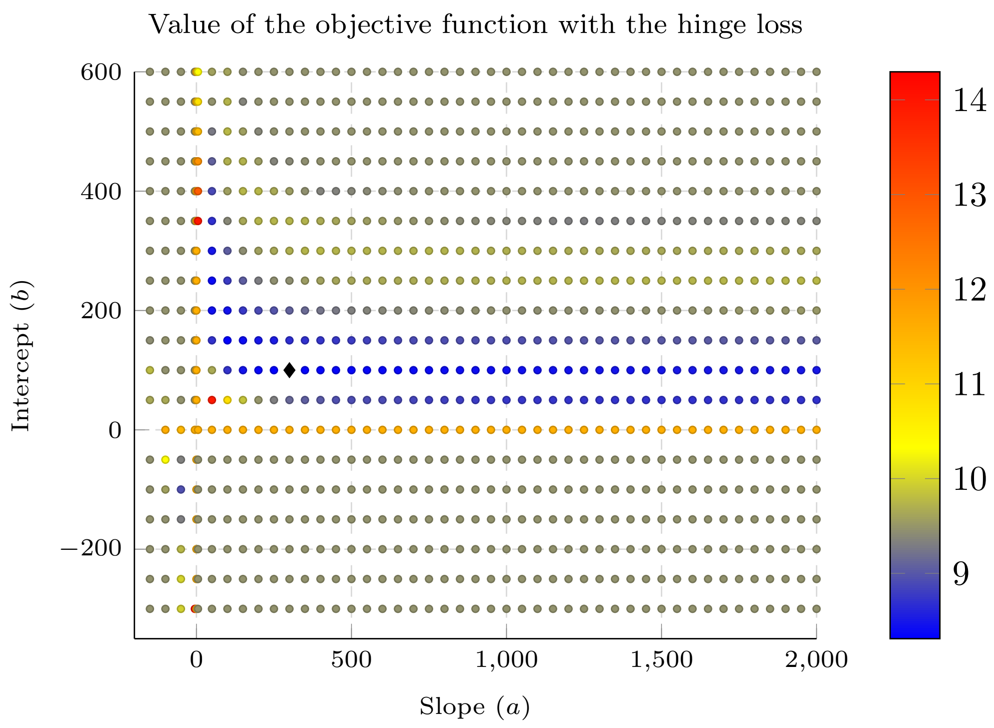

Figure 7.2: The estimated value of the objective function in the third class. The black diamond indicates the lowest value. The ratio between corrective and preventive maintenance is 10, $k = 2$ and $\alpha = 0.8$.

Table 7.1: The thresholds and parameters for the three policy classes for different cost ratios between corrective and preventive replacements. We use $\alpha = 0.8$ and $k = 2$. A "-" means that there is no real solution to the optimization problem.

Ratio	**Loss**	**Class 1** (WH, D)	**Class 2** (η)	**Class 3** (a, b)
2	hinge	(-, 349)	0.03	(550, 400)
2	exponential	(-, 349)	0.03	(900, 400)
5	hinge	(13230, 182)	0.03	(550, 400)
5	exponential	(13412, 190)	0.03	(900, 400)
10	hinge	(7962, 131)	0.89	(300, 100)
10	exponential	(8003, 134)	0.03	(900, 400)
50	hinge	(2874, 52)	0.94	(150, 50)
50	exponential	(2876, 51)	0.94	(150, 50)
100	hinge	(1893, 32)	0.96	(150, 50)
100	exponential	(1894, 32)	0.96	(150, 50)

Chapter 8

Comparison Of Policies

In Chapter 7 we discuss how solutions of the main optimization problem can be numerically approximated. We see that there are big differences in the policies, depending on the relative weight of the precision. Lower weights result in optimal policies with many replacements, whereas higher weights favor conservative policies. The current SVM based policy is also conservative and in this chapter we compare the old and the new policies.

We compare the policies in two ways. We first look at the training set used to train the SVM to see how the new policies, see Table 7.1, perform when judged purely on their classification ability. After that, we perform a simulation study. In the study, we simulate monitors and investigate the influence of some policies on the cost, expected usage time per monitor, i.e. cycle length, and the number of preventive replacements.

8.1 Training Set Comparison

To get a first indication on how the policies perform, we look at the training set for the SVM. On the training set, we compare the classification performance of the new and old policies. For this purpose, we use the new policies, found in Chapter 7, purely as a classifier, so data entries that fall in the region where a preventive replacement is prescribed are classified as failed. We compare the performance with the SVM trained on all eleven features, but also look at an SVM which only uses features 1,2 and 5. In this way, we can see how much the SVM looses in predictive power when only three features are used.

To compare the policies we investigate how well they classify the data, which we measure with the accuracy, precision, recall and false positive rate (FPR) measures, see Table 8.1 and [11]. All the measures are based on the confusion matrix of a classification, which specifies the number of true positives (`TP`), false positives (`FP`), true negatives (`TN`) and false negative (`FN`). The positives in the training set are the failures and the negatives are thus the functioning monitors.

We compute the measures in similar way to the SHAP values in Section 2.3, see Algorithm 8.1. We divide the training set into ten equally large set and use each set once for classification and nine times for training the SVMs.

Table 8.1: The definition of the performance measures that we use to compare the classifiers.

Measure	Formula	Description
Accuracy	$\frac{\texttt{TN}+\texttt{TP}}{\texttt{TN}+\texttt{TP}+\texttt{FN}+\texttt{FN}}$	Fraction of correctly classified instances.
Precision	$\frac{\texttt{TP}}{\texttt{TP}+\texttt{FP}}$	Fraction of positive predicted instances that are positive.
Recall	$\frac{\texttt{TP}}{\texttt{TP}+\texttt{FN}}$	Fraction of positive instances that are predicted positive.
FPR	$\frac{\texttt{FP}}{\texttt{TN}+\texttt{FP}}$	Fraction of positive predicted instances that are negative.

Algorithm 8.1: Procedure to determine the performance of several policies on the training set using 10-fold cross validation using stratified sampling as in Algorithm 2.1.

```
   Data: Training set (TS), 10 sets of indices as obtained in Algorithm 2.1
   Result: Classification performance of policies
1  for $i$ in 1 to 10 do
2      Test set is the subset of TS consisting of indices in set $i$
3      Training set is the rest of TS
4      Train an SVM on training set
5      Train an SVM on training set, only using features 1,2 and 5
6      Classify the points in the test set using the SVM classifiers
7      Classify the points in the test set using the new policies as classifier
8      Compute the accuracy, precision, recall and FPR for each classification
9  Average the accuracy, precision, recall and FPR for each policy
```

In Table 8.2 the results are shown. We see that the more conservative class 3 policies, i.e. the policies $(a, b) = (550, 400)$ and $(a, b) = (900, 400)$, have a similar accuracy and precision as the SVM classifier. For this reason, we will use these policies as substitutes for the SVM in the next section. One other noticeable observation is that the performance of an SVM trained on the three features used to develop the degradation process is outperformed by several of the policies obtained by solving Equation (1.1).

Table 8.2: The classification results on the training set when the policies are used as classifiers (see Algorithm 8.1 for the procedure). We train two SVM classifiers: one using eleven features and one using three features, denoted by SVM-3. A common practice is to use the Platt scaling to obtain class probabilities out of the classifier, instead of only a label. The class probabilities are then used to classify the data. For instance, SVM-Platt - 0.3 classifies points with a class probability of being normal that are larger than 0.3 as normal.

Policy	**Accuracy**	**Precision**	**Recall**	**FPR**
Class 1 - (13412, 190)	0.7106	0.6775	0.2063	0.0482
Class 1 - (1893, 32)	0.3614	0.3218	0.8805	0.8869
Class 1 - (2874, 52)	0.4030	0.3277	0.8055	0.7896
Class 1 - (7962, 131)	0.6189	0.3997	0.3448	0.2498
Class 1 - (8003, 134)	0.6174	0.3948	0.3307	0.2454
Class 2 - 0.03	0.6924	0.9157	0.0564	0.0034
Class 2 - 0.89	0.4508	0.3293	0.6746	0.6563
Class 2 - 0.94	0.3947	0.3271	0.8244	0.8109
Class 2 - 0.96	0.3780	0.3264	0.8688	0.8567
Class 3 - (150, 50)	0.4008	0.3247	0.7916	0.7862
Class 3 - (300, 100)	0.6280	0.4318	0.4803	0.3012
Class 3 - (550, 400)	0.6864	0.8812	0.0376	0.0034
Class 3 - (900, 400)	0.6864	0.8812	0.0376	0.0034
SVM-Hyper	0.6977	0.9008	0.0730	0.0034
SVM-Platt - 0.3	0.6955	0.9008	0.0659	0.0034
SVM-Platt - 0.5	0.7038	0.8148	0.1293	0.0212
SVM-Platt - 0.6	0.6894	0.6388	0.2165	0.0838
SVM-Platt - 0.7	0.5212	0.3879	0.7279	0.5781
SVM-3-Hyper	0.6864	0.8812	0.0376	0.0034
SVM-3-Platt - 0.3	0.6864	0.8812	0.0376	0.0034
SVM-3-Platt - 0.5	0.6864	0.8812	0.0376	0.0034
SVM-3-Platt - 0.6	0.6864	0.8812	0.0376	0.0034
SVM-3-Platt - 0.7	0.3235	0.3235	1.0000	1.0000

8.2 Simulation Study

The comparison in Section 8.1 does not allow us to consider the costs. We therefore simulate monitors and investigate the performance of the policies on the simulated data.

In Section 7.2 we discuss how we simulate monitors (Algorithm 7.2) and how we estimate the objective function (Algorithm 7.3). We are now interested in the cost, expected usage time per monitor and the number of preventive replacements. We can therefore almost use the same procedure as outlined in Algorithm 7.3, but, instead of the objective function, we calculate the individual performance indicators.

To obtain confidence intervals we use a bootstrapping procedure outlined in Section 8 of [35], see Algorithm 8.2. We construct bootstrap values by taking a random subset of the N sample paths and calculate the quantities based on the subset. Define z_0 as $z_0 = \Phi^{-1}(\widehat{G}(\widehat{\xi}))$ (Equation (7.8) in [35]), where Φ^{-1} is the inverse of the standard normal distribution, $\widehat{\xi}$ the estimated quantity, e.g. the average cost, and $\widehat{G}$ the empirical cumulative distribution function of the quantity obtained from the bootstraps. The confidence interval for the quantity is then given by $\widehat{G}^{-1}(\Phi(2z_0 \pm z_\alpha))$ (Equation (7.9) in [35]), where z_α is the α-quantile of the standard normal distribution.

Algorithm 8.2: Pseudo code for the bootstrapping procedure used to obtain a confidence interval for the average cost.

```
Data: age and cost vectors of the $N$ simulated paths (see Algorithm 7.3), $N_B$
      the number of bootstraps, $B$ the number of samples per bootstrap
Result: Confidence interval of the average cost
1 for $i$ in 1 to $N_B$ do
2     Pick $B$ values from 1 to $N$ (with replacement), $a_1, a_2, \ldots, a_B$
3     avg.cost[i] = $\frac{\sum_{k=1}^{B} cost[a_k]}{\sum_{k=1}^{B} time[a_k]}$
4 $\widehat{\xi} = \frac{\sum_{i=1}^{N} cost[i]}{\sum_{i=1}^{N} age[i]}$
5 $z_0 = \Phi^{-1}(\widehat{G}(\widehat{\xi}))$
6 Confidence interval is $\widehat{G}^{-1}(\Phi(2z_0 \pm z_\alpha))$
```

In the simulation study we want to consider the same policies as in Section 8.1. However, since we only simulate three features, we cannot use the SVM trained on eleven features on the simulated data. As a replacement we use the conservative classifiers of class three, which performed similar on the training set, see Table 8.2.

In the simulation we use 10^5 simulated monitors. The results are shown in Tables 8.3 and 8.4. We see that many policies have low precision. We discuss the reason for the low precision in Section 7.2, where we state that for $k = 2$ the cost dominates the objective function. Another reason for the low precision is the strict definition of a correct replacement. Only replacements that take place 30 days before the failure, where each day is a working day of ten hours, are classified as correct replacements.

In Table 8.4 we show the costs of the different policies. In general, we see that policies with a high precision have low costs when the ratio is low. However, when the cost ratio

Table 8.3: The performance of policies in Table 7.1. The columns show the number of preventive replacements (# PM), the number of correct replacements, then number of corrective replacements (# CM), the precision as calculated in Equation (6.1) and the expected time until replacement ($\mathbb{E}[Z]$). The mean time to failure of the monitors equals 14656 hours. The results are based on 10^5 simulated monitors.

Policy	**#PM**	**# Correct**	**#CM**	**Precision**	$\mathbb{E}[Z]$
Class 1 - (13412, 190)	52601	2874	47399	0.0546	10507
Class 1 - (1893, 32)	96578	857	3422	0.0089	1819
Class 1 - (2874, 52)	93770	1000	6230	0.0107	2781
Class 1 - (7962, 131)	75012	1781	24988	0.0237	7100
Class 1 - (8003, 134)	74846	1790	25154	0.0239	7132
Class 2 - 0.03	8431	6606	91569	0.7835	14616
Class 2 - 0.89	89166	1346	10834	0.0151	4158
Class 2 - 0.94	94149	1110	5851	0.0118	2742
Class 2 - 0.96	96145	1036	3855	0.0108	2061
Class 3 - (150, 50)	93431	1225	6569	0.0131	2995
Class 3 - (300, 100)	75713	2212	24287	0.0292	7057
Class 3 - (550, 400)	7264	6715	92736	0.9244	14649
Class 3 - (900, 400)	6357	6355	93643	0.9997	14656

increases, avoiding corrective replacements becomes more attractive and the best policies from a cost perspective are the ones with lower precision. Already at a cost ratio of 5, we see a cost decrease of fourteen percent between the best policy and the SVM substitutes. At a cost ratio of ten, the decrease is already more than 28%. The decrease with the other policies that have a high precision is minimal.

Another noticeable observation is that the policies of class 1, which we approximated by the univariate thresholds and were the computationally easiest to obtain, have the lowest costs for cost ratios of 5 and 10. For most cost ratios, the differences between the optimal policies per class are small, especially compared to the SVM substitutes.

To investigate the accuracy of the estimates, we look at the confidence interval of a few estimates. We look at the following policies: policy $(\tau_{WH}, \tau_D) = (7962, 131)$ from class 1, policy $\eta = 0.89$ from class 2 and $(a, b) = (550, 400)$ from class 3. We use the procedure in Algorithm 8.2 to construct 95% confidence intervals, we use $N_B = 5000$ and $B = 20,000$.

The confidence intervals are shown in Table 8.5. We see that the intervals of the class 1 and class 2 policies do not overlap with the SVM substitute (class 3 policy). We therefore conclude that the decrease in cost is significant. If we look at the upper bound for the class 1 and 2 policies and the lower bound of the class 3 policy, we see a decrease in costs of 27% and 24%, respectively.

In Table 8.3 we see that the lifetimes of the monitors are utilized well and that the precision is high for the conservative policies, including the SVM substitute. These

Table 8.4: The precision and average cost criterion of the policies in Table 7.1 for several cost ratios. The results are based on 10^5 simulated monitors.

Policy	**Precision**	**Ratio 2**	**Ratio 5**	**Ratio 10**	**Ratio 50**	**Ratio 100**
Class 1 - (13412, 190)	0.0546	2.0560	4.0395	7.3452	33.7911	66.8486
Class 1 - (1893, 32)	0.0089	8.3316	9.1586	10.5369	21.5639	35.3475
Class 1 - (2874, 52)	0.0107	5.5975	6.5823	8.2237	21.3546	37.7682
Class 1 - (7962, 131)	0.0237	2.5801	4.1276	6.7067	27.3398	53.1311
Class 1 - (8003, 134)	0.0239	2.5719	4.1227	6.7073	27.3841	53.2301
Class 2 - 0.03	0.7835	1.9210	4.6756	9.2666	45.9949	91.9054
Class 2 - 0.89	0.0151	3.9063	5.0518	6.9610	22.2345	41.3264
Class 2 - 0.94	0.0118	5.6576	6.5957	8.1594	20.6685	36.3048
Class 2 - 0.96	0.0108	7.3852	8.2076	9.5782	20.5434	34.2500
Class 3 - (150, 50)	0.0131	5.2146	6.1789	7.7861	20.6434	36.7150
Class 3 - (300, 100)	0.0292	2.5811	4.0943	6.6162	26.7913	52.0102
Class 3 - (550, 400)	0.9244	1.9283	4.7117	9.3508	46.4631	92.8535
Class 3 - (900, 400)	0.9997	1.9364	4.7457	9.4279	46.8851	93.7066

Table 8.5: Confidence intervals on the estimated average costs for a cost ratio of 10.

Policy	Lower	Estimate	Upper
Class 1 - (7962, 131)	6.6064	6.7074	6.8096
Class 2 - 0.89	6.8299	6.9620	7.0938
Class 3 - (550, 400)	9.3168	9.3501	9.3812

goals of Philips are thus met. However, looking at the cost aspect of the policies, the conservative policies are far worse.

Chapter 9

Relation to SVMs

The starting point of the project was the SVM based predictive model. We believed that improvements could be made, if the operational costs were taken into account. We decided to not extend the SVM formulation to consider the costs, but instead to set up a stochastic optimization problem. The objective function of the optimization problem incorporates two performance indicators: the operational costs and a function of the precision. So far, we saw that the solution space of the new method is larger than the solutions space of the old method (Chapter 4) and that the new method gives solutions that outperform, at least cost-wise, the old solutions (Chapter 8). The remaining question is whether we could have obtained similar results, if we had tried to extend the SVM instead of setting up an optimization problem.

In this chapter we discuss some recent papers that extend the SVM in some way and discuss why the methods of the papers cannot directly be applied in our setting. We conclude that the current methods do not allow us to include the two performance indicators in the SVM formulation to obtain a data-driven machine learning problem.

In the introduction we mention that including the operational costs in the decision-making process could significantly lower the costs. The reason for this is that a classifier based on the SVM algorithm, which does not take any costs into account, is used for decision-making, where costs play a important role. In [29] they call this the sequential process. The authors propose a simultaneous process, where a loss on the training data, e.g. hinge loss or exponential loss, and some operational costs are combined in the objective function. The setting in the paper is a general machine learning setting and their definition of operational costs depends on the problem. For example, they use the total costs of buying and repairing houses, and the total time needed to perform a certain task as operational costs.

The difficulty in applying their approach to our setting lies with the definition of the operational costs. To use the average cost criterion as definition enforces us to accurately determine the moment of replacement, for which we need the whole sample path. Including whole sample paths in the training set results in a large set which leads to computational difficulties. To overcome the need of whole paths, we could use another cost definition. However, the alternative definition may not include the cycle length, which, as we discuss in Chapter 5, can lead to unwanted solutions.

A way to consider the costs after training a set of classifiers is described in [28]. In the paper the authors take the operating conditions into considerations by assigning costs to erroneous classifications. These costs are then used to find the classifier with the lowest cost. The approach in [28] is still a sequential process and the costs only consider erroneous classifications, which would be faulty in our case.

The study in [36] extends the objective function of the SVM algorithm to include certain common performance measures, such as the accuracy and true positive rate. Similar to the target α in the penalty functions in Chapter 6, the authors of [36] have a target value. However, they include the target as a hard constraint, meaning that the solution must satisfy the target. The advantage of this approach is that one knows that a solution will satisfy the constraints. Disadvantages of the method include an increase in the complexity and that the feasibility is only guaranteed if kernels are used. The use of kernels is a common practice for SVMs, but the results are less interpretable and therefore unwanted in the development of preventive maintenance policies. Just as the study introducing the simultaneous process ([29]), the study focuses on non-sequential data. To adapt the results to the setting of this report, would require whole degradation paths to be included in the training set, which increases the computational time needed to solve the problem even more.

Another possible building block to add the operational costs and precision in an SVM setting are the knowledge based SVMs, which incorporate prior knowledge in the classifier, see for instance [37, 38]. One possibility to use the techniques of knowledge based SVMs is to include an area in the working hour-degradation plane where a preventive replacement may not be performed. The area can be defined by domain experts, which can increase the interpretability and trust of the found policy. Furthermore, only failures inside this area need to be included in the training set, since they are needed to determine the precision. For working monitors no data points in the area are needed, which reduces the data points in the training set.

In this chapter we discuss some advancements in the literature regarding extensions of the classical SVM formulation. The main problem with the extensions are that they focus on including either some form of operational costs or performance measures, but not both, and that they are developed for non-sequential data, which makes it difficult to apply them in our setting. As the literature develops, the main optimization problem of Equation (1.1) might be formulated as a data-driven machine learning problem.

Chapter 10

Conclusion

10.1 Conclusion

In this report, we introduce an optimization problem in order to find a preventive maintenance policy for the big screen monitors of the iXR systems. The objective function of the optimization problem takes the operational costs and a performance measure into consideration.

We start by ranking the features of a current predictive model to find the features with the most predictive power. The topped ranked features are used to build a stochastic process describing the degradation of a monitor. In the subsequent chapters we introduce the average costs, which we define as the cost per time unit, and penalty functions for the precision, which are based on common loss functions in machine learning literature. To reduce the complexity of the optimization problem, we introduce three classes of policies with desired properties. We show how to solve the optimization problem and in a simulation study show that we can achieve a significant decrease in the costs when using a policy favoring low operational costs. For instance, in a scenario with a cost ratio of 10, the decrease in cost is more than 28%. For policies that favor a high precision there is little to no decrease in the average costs. From this we conclude that it seems beneficial to convince customers to allow more preventive replacements for high valued components.

10.2 Discussion

Two observations that one can make about Figure 3.2 and generally hold true for the monitors under consideration (see also next point) were not discussed in Chapter 3, namely the fluctuations of the measurements and the large gap. The large fluctuations in the brightness measurements make it necessary to come up with some definition of the "true" brightness. We chose to represent the brightness of a monitor with the third running minimum for reasons explained in Section 3.3. The assumption behind this is that the lower values of the brightness give a better indication of the state than the others. However, there are some indications that this might not be true. For instance, it seems that measurements made after a restart during the day often are higher than the measurements at the beginning of said day, which could indicate that the monitors need more time to reach the actual brightness level than the time between turning the monitor on and the measurement. Therefore, one could argue that the higher values give more information about the state. The large gaps in the data influence the estimations of the parameters of the jump process, since it is likely that there is a large jump after a large gap.

In this report we use the replacement of the big screens as vehicle of illustration, specifically the old version of the big screen. Newer screens are used in the same way as the old screens, but there are some changes in the specifications. Most notable is that the newer screens have little to no fluctuations in the brightness measurements and that it takes longer before the measurements start to decrease. As a result, the proposed model will not work on the new monitor versions. A possible adaption of the model is to include an exceptional first time, meaning that the time until the first jump is differently distributed than the other interarrival times.

In Section 3.4.1 we note that the estimates for the parameters of the interarrival times and jumps size vary between the monitors. The variation between the monitors is not taken into consideration in this project. In future research the fixed parameters could instead be random to better represent the whole pool of monitors. To make decisions, one then needs to estimate the parameters while the monitor is operating. One way to do this is to start with some parameters, e.g. the ones used in this report, and then use Bayesian updating to improve the estimates. The updating will have the largest impact on monitors that degrade slowly, since, for these monitors, the updated estimates will result in policies that preventively replace the monitor later, thus utilizing more of the monitor's lifetime.

The data used to estimate the distribution of L, D and E was gathered to train an SVM model. In the maximum likelihood estimation procedure it is important that the data is independent. We could therefore not use all the data in the training set, since a large portion of the instances labeled as failed are dependent. The removal of the dependent instance makes the data more unbalanced, resulting in a ratio between normal and failure instance of 8 to 1. Collecting more data on failed machines and data on the reason of failure could result in different and more accurate estimations.

In Chapters 4 and 7 we look into three classes of policies and find the optimal policy in each class. The classes are selected because of their simplicity and interpretational value, but there is not guarantee that the optimal policy is a member of one of the three classes. Further research could focus on other policy classes and optimality theorems.

In Chapter 3 we rank the features used in an SVM model to select the three features with the most predictive power. The degradation process is then constructed to take the three features into consideration. Reducing the number of features will lead to a loss in predictive power. Including more variables in the stochastic model, without losing interpretability is another question to be answered. In the setting of the big screen monitors, we saw that, on the training set, policies based on the three features performed better, but there is no guarantee that this is the case in other scenarios. On the contrary, it is likely that the performance decreases.

Bibliography

[1] R. Sipos, D. Fradkin, F. Moerchen, and Z. Wang, “Log-based predictive maintenance,” in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’14, (New York, NY, USA), pp. 1867–1876, ACM, 2014.

[2] D. J. White, “A survey of applications of markov decision processes,” *Journal of the Operational Research Society*, vol. 44, no. 11, pp. 1073–1096, 1993.

[3] X. Wu and V. Kumar, *The Top Ten Algorithms in Data Mining.* Chapman & Hall/CRC, 1st ed., 2009.

[4] C. J. C. Burges, “A Tutorial on Support Vector Machines for Pattern Recognition,” *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[5] D. Meyer, E. Dimitriadou, K. Hornik, A. Weingessel, and F. Leisch, *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*, 2017. R package version 1.6-8.

[6] S. M. Lundberg and S.-I. Lee, “A unified approach to interpreting model predictions,” *Advances in Neural Information Processing Systems*, pp. 4765–4774, 2017.

[7] R. B. Myerson, *Game theory - Analysis of Conflict.* Harvard University Press, 1997.

[8] C. Molnar, B. Bischl, and G. Casalicchio, “iml: An r package for interpretable machine learning,” *The Journal of Open Source Software*, vol. 3, no. 26, p. 786, 2018.

[9] E. Strumbelj and I. Kononenko, “An efficient explanation of individual classifications using game theory,” *Journal of Machine Learning Research*, vol. 11, p. 1–18, 2010.

[10] B. H. Richard M. Heiberger, *Statistical Analysis and Data Display. An Intermediate Course with Examples in R. Second Edition.* Springer, 2015.

[11] M. Sokolova and G. Lapalme, “A systematic analysis of performance measures for classification tasks,” *Information Processing and Management*, vol. 45, no. 4, pp. 427–437, 2009.

[12] A. Wald and J. Wolfowitz *The Annals of Mathematical Statistics.*

[13] G. M. Ljung and G. E. P. Box, "On a measure of lack of fit in time series models," *Biometrika*, vol. 65, no. 2, pp. 297–303, 1978.

[14] P. Brockwell and R. Davis, *Introduction to Time Series and Forecasting.* Springer Texts in Statistics, Springer International Publishing, 2016.

[15] F. Caeiro and A. Mateus, *randtests: Testing randomness in R*, 2014. R package version 1.0.

[16] R. Heller, Y. Heller, and M. Gorfine, "A consistent multivariate test of association based on ranks of distances," *Biometrika*, vol. 100, no. 2, pp. 503–510, 2013.

[17] B. B. . S. Kaufman, based in part on an earlier implementation by Ruth Heller, and Y. Heller., *HHG: Heller-Heller-Gorfine Tests of Independence and Equality of Distributions*, 2017. R package version 2.2.

[18] F. Abramovich and Y. Ritov, *Statistical Theory. A Concise Introduction.* Chapman and Hall/CRC, 2013.

[19] P. Hougaard, *Analysis of Multivariate Survival Data.* Springer, 2000.

[20] J. G. MacKINNON, "Bootstrap methods in econometrics," *Economic Record*, vol. 82, no. s1, pp. S2–S18, 2006.

[21] M. L. Delignette-Muller and C. Dutang, "fitdistrplus: An R package for fitting distributions," *Journal of Statistical Software*, vol. 64, no. 4, pp. 1–34, 2015.

[22] C. Suijkerbuijk, "Integration of preventive maintenance and inventory management for healthcare systems at philips healthcare," Master's thesis, Eindhoven University of Technology, 2017. `https://pure.tue.nl/ws/portalfiles/portal/89095742/0808814_thesis_Corn_Suijkerbuijk_PUBLIC_VERSION.pdf`.

[23] M. X. C. D. Lai and D. N. P. Murthy, "A modified weibull distribution," *IEEE Transactions on reliability*, vol. 52, pp. 33–37, 2003.

[24] U. Hjorth, "A reliability distribution with increasing, decreasing, constant and bathtub-shaped failure rates," *Technometrics*, vol. 22, pp. 99–107, 1980.

[25] M. Xie and C. D. Lai, "Reliability analysis using an additive weibull model with bathtub-shaped failure rate function," *Reliability Engineering & System Safety*, vol. 52, pp. 87–93, 1996.

[26] E. Kaplan and P. Meier, "Nonparametric estimation from incomplete observations," *Journal of American Statistical Association*, vol. 53, pp. 457–481, 1958.

[27] T. M. Therneau, *A Package for Survival Analysis in S*, 2015. version 2.38.

[28] J. Korst, V. Pronk, M. Barbieri, and S. Consoli, *Data Science for Healthcare: Methodologies and Applications*, ch. Introduction to Classification Algorithms and their Performance Analysis using Medical Examples., pp. 39–73. Springer, 2019.

[29] T. Tulabandhula and C. Rudin, "Machine learning with operational costs," *Journal of Machine Learning Research*, vol. 14, pp. 1989–2028, 2013.

[30] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* New York, NY, USA: John Wiley & Sons, Inc., 1st ed., 1994.

[31] S. M. Ross, *Introduction to Probability Models.* San Diego, CA, USA: Academic Press, tenth ed., 2010.

[32] S. Ross, *Stochastic Processes.* Wiley series in probability and statistics: Probability and statistics, Wiley, 1996.

[33] "Some recent results on the distributions of stopping times of compound poisson processes with linear boundaries," *Journal of Statistical Planning and Inference*, vol. 130, no. 1, pp. 95 – 109, 2005.

[34] M. Nyberg, T. Ambjörnsson, and L. Lizana, "A simple method to calculate first-passage time densities with arbitrary initial conditions," *New Journal of Physics*, vol. 18, p. 063019, June 2016.

[35] B. Efron and R. Tibshirani, "Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy," *Statistical Science*, vol. 1, no. 1, pp. 54–75, 1986.

[36] E. C. P. R.-C. Sandra Benítez-Peña, Rafael Blanquero, "On support vector machines under a multiple-cost scenario," *Advances in Data Analysis and Classification*, pp. 1–20, 2018.

[37] Q. V. Le, A. J. Smola, and T. Gärtner, "Simpler knowledge-based support vector machines," in *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, (New York, NY, USA), pp. 521–528, ACM, 2006.

[38] G. M. Fung, O. L. Mangasarian, and J. W. Shavlik, "Knowledge-based support vector machine classifiers," in *Proceedings of the 15th International Conference on Neural Information Processing Systems*, NIPS'02, (Cambridge, MA, USA), pp. 537–544, MIT Press, 2002.

List of Figures

List of Tables

List of Algorithms

Estimates Interarrival Times and Jump Sizes

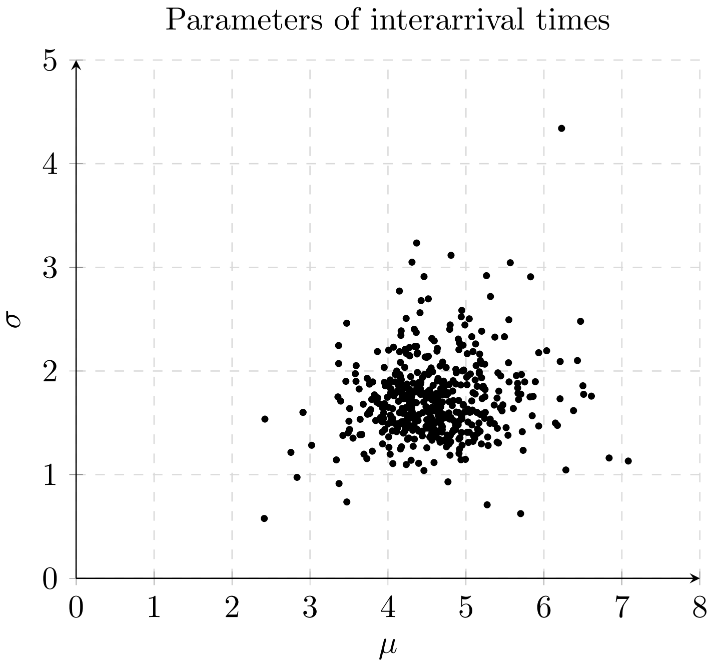

Figure A.1: The estimated parameters of the lognormal distribution for the interarrival time.

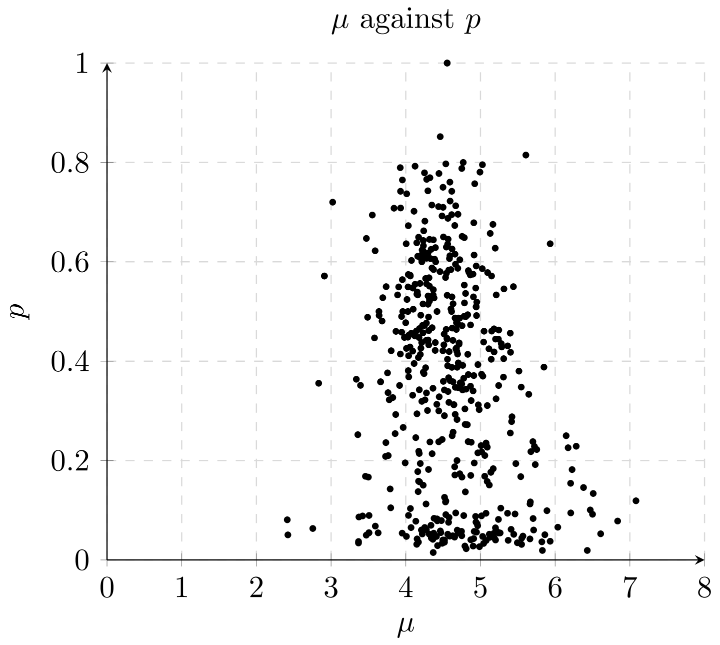

Figure A.2: The estimated μ parameter of the lognormal distribution for the interarrival time versus the estimated p parameter in the geometric distribution.

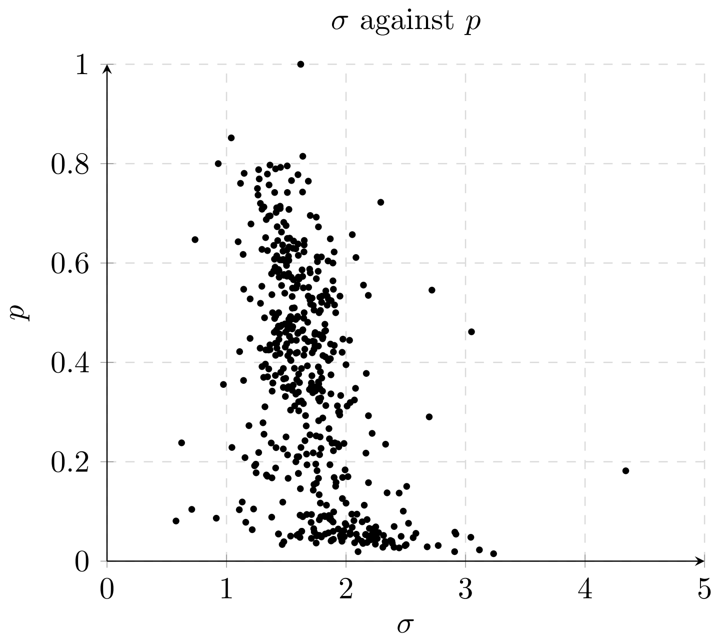

Figure A.3: The estimated σ parameter of the lognormal distribution for the interarrival time versus the estimated p parameter in the geometric distribution.

Appendix B

Independence Tests

B.1 Wald-Wolfowitz

The Wald-Wolfowitz runs test has the following hypothesis:

H_0 The sequence was produced in a random way.

H_A The sequence was not produced in a random way.

The test is performed in the following way.

Let $x_1, x_2, \ldots, x_n$ be continuous observations with no ties and let m be the median of the observations. Define binary variables $\widetilde{x_1}, \widetilde{x_2}, \ldots, \widetilde{x_n}$ as

$$\widetilde{x_i} = \begin{cases} 1, & \text{if } x_i \geq m \\ 0, & \text{if } x_i < m. \end{cases}$$

A run is defined as a sequence consisting of the same number, so either a sequence of zeros or a sequence of ones. For instance, the data sequence $1, 0, 0, 1, 1$ has three runs. Let N_0 be the number of runs consisting of zeros and N_1 the number of runs consisting of ones.

For large sample sizes, the number of runs can be approximated by a normal distribution with mean μ, given by

$$\mu = \frac{2N_0N_1}{N_0 + N_1} + 1,$$

and variance σ^2, given by

$$\sigma^2 = \frac{2N_0N_1(N_0N_1 - N_0 - N_1)}{(N_0 + N_1)^2(N_0 + N_1 + 1)}.$$

We reject the null hypothesis at significance level α if

$$w = \frac{N_0 + N_1 - \mu}{\sigma}$$

is larger than $z_{1-\alpha/2}$ or smaller than $-z_{1-\alpha/2}$, where $z_{1-\alpha/2}$ is the $1 - \alpha/2$ quantile of the standard normal distribution.

B.2 Ljung-Box

The Ljung-Box test is defined as follows:

H_0 The data are independently distributed.

H_A The data are not independently distributed; they exhibit serial correlation.

Let $x_1, x_2, \ldots, x_n$ be continuous observations. The test statistic for the Ljung-Box test is given by

$$Q = n(n+2)\sum_{j=1}^{h} \frac{\widehat{\rho}^2(j)}{n-j},$$

where $\widehat{\rho}(j)$ is the sample autocorrelation at lag j and h is the number of lags that are tested. The sample autocorrelation is given by

$$\widehat{\rho}(j) = \frac{\widehat{\gamma}(j)}{\widehat{\gamma}(0)} \text{ for } -n < j < n,$$

where $\widehat{\gamma}(j)$ is given by

$$\widehat{\gamma}(j) := \frac{1}{n}\sum_{t=1}^{n-j}(x_{t+j} - \bar{x})(x_t - \bar{x}) \text{ for } -j < j < j,$$

and $\bar{x}$ is the sample mean, given by

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i.$$

The test statistic is approximated by the chi-squared distribution with h degrees of freedom. The null hypothesis is rejected at confidence level α if $Q > \chi^2_{1-\alpha}(h)$, where $\chi^2_{1-\alpha}(h)$ is the $1-\alpha$ quantile of the chi-squared distribution with h degrees of freedom.

B.3 Difference-sign test

The difference-sign test is defined as follows:

H_0 There is no trend in the data.

H_A There is either an increasing or a decreasing trend in the data.

The test statistic for the difference-sign test is S, which is number of times that a data point is larger than its predecessor, i.e. the number of values i such that $x_i > x_{i-1}$, $i = 2, 3, \ldots, n$ for a sequence $(x)_{i=1,2,\ldots,n}$.

Under the null hypothesis, the expected value of S is given by $\mu_S = \frac{1}{2}(n-1)$ and the variance by $\sigma_S^2 = \frac{n+1}{12}$. For large n, it holds that S is approximately normal with mean μ_S and variance σ_S^2. We reject the null hypothesis at level α if $\frac{S-\mu_S}{\sigma_S} > z_{1-\alpha/2}$, where $z_{1-\alpha/2}$ is the $1-\alpha/2$ quantile of the standard normal distribution.

Proof of Theorem 7.1

(Proof of Theorem 7.1). For a given threshold $\boldsymbol{\sigma}$, using the general relation

$$R = c_p \mathbb{1}\{S \leq T\} + c_c \mathbb{1}\{S > T\}$$

and the fact that S is now a deterministic random variable, given by $\boldsymbol{\sigma}$, we have that

$$\mathbb{E}[R] = \mathbb{E}[c_c \mathbb{1}\{\min\{T, \boldsymbol{\sigma}\} = T\} + c_p \mathbb{1}\{\min\{T, \boldsymbol{\sigma}\} = \boldsymbol{\sigma}\}] = c_c \mathbb{P}(T \leq \boldsymbol{\sigma}) + c_p \mathbb{P}(T \geq \boldsymbol{\sigma}) \tag{C.1}$$

and

$$\mathbb{E}[Z] = \mathbb{E}[\min\{T, \boldsymbol{\sigma}\}] = \int_{t=0}^{\boldsymbol{\sigma}} t f(t)\,\mathrm{d}t + \boldsymbol{\sigma}\mathbb{P}(T \geq \boldsymbol{\sigma}) = \int_0^{\boldsymbol{\sigma}} 1 - F(t)\,\mathrm{d}t\,. \tag{C.2}$$

For the last equality in Equation (C.2) we use integration by parts. Equation (7.3) is obtained by inserting Equation (C.1) and Equation (C.2) in Equation (5.2).

To find the minimum of $g(\boldsymbol{\sigma})$, we take the derivative with respect to $\boldsymbol{\sigma}$ and get

$$g'(\boldsymbol{\sigma}) = \frac{1 - F(\boldsymbol{\sigma})}{\int_0^{\boldsymbol{\sigma}} 1 - F(t)\,\mathrm{d}t}\left[(c_c - c_p)h(\boldsymbol{\sigma}) - g(\boldsymbol{\sigma})\right].$$

We cannot get a closed expression for the $\boldsymbol{\sigma}$ at which $g'(\boldsymbol{\sigma}) = 0$, but we know that, for $\boldsymbol{\sigma}$ such that $g'(\boldsymbol{\sigma}) = 0$, either $(c_c - c_p)h(\boldsymbol{\sigma}) = g(\boldsymbol{\sigma})$ or $1 - F(\boldsymbol{\sigma}) = 0$. The second case is not interesting, because the threshold is then above the maximum lifetime of the component, which. We then follow a corrective maintenance policy.

Now assume that we have found a $\boldsymbol{\sigma}$ such that $g'(\boldsymbol{\sigma}) = 0$ and $(c_c - c_p)h(\boldsymbol{\sigma}) = g(\boldsymbol{\sigma})$. For the second derivative, we get that

$$\begin{aligned} g''(\boldsymbol{\sigma}) = {} & \frac{1 - F(\boldsymbol{\sigma})}{\int_0^{\boldsymbol{\sigma}} 1 - F(x)\,\mathrm{d}x}\Big[(c_c - c_p)h'(\boldsymbol{\sigma}) - g'(\boldsymbol{\sigma})\Big] \\ & + \frac{-\int_0^{\boldsymbol{\sigma}} 1 - F(x)\,\mathrm{d}x\, f(\boldsymbol{\sigma}) - (1 - F(\boldsymbol{\sigma}))(1 - F(\boldsymbol{\sigma}))}{(\int_0^{\boldsymbol{\sigma}} 1 - F(x)\,\mathrm{d}x)^2}\Big[(c_c - c_p)h(\boldsymbol{\sigma}) - g(\boldsymbol{\sigma})\Big]. \end{aligned} \tag{C.3}$$

Using the fact that $g'(\boldsymbol{\sigma}) = 0$ and $(c_c - c_p)h(\boldsymbol{\sigma}) = g(\boldsymbol{\sigma})$, Equation (C.3) reduces to

$$g''(\sigma) = \frac{1 - F(\boldsymbol{\sigma})}{\int_0^{\boldsymbol{\sigma}} 1 - F(t)\,\mathrm{d}t} \cdot (c_c - c_p)h'(\boldsymbol{\sigma}).$$

We see that if $h'(\boldsymbol{\sigma})$ is negative then $g''(\boldsymbol{\sigma})$ is negative and we thus have a maximum. Therefore, it is not worthwhile to have a preventive maintenance policy. When $h'(\boldsymbol{\sigma})$ is positive, $g''(\boldsymbol{\sigma})$ is positive and thus we obtain a minimum. In this case minimal costs are obtained when preventive maintenance is applied. $\square$