# A data-driven decision support model for planned maintenance reduction in healthcare

van Lohuizen, T.

*Award date:*
2018

# A Data-Driven Decision Support Model for Planned Maintenance Reduction in Healthcare

By

Tim van Lohuizen
Student Number: 0812808
t.v.lohuizen@student.tue.nl

in partial fulfilment of the requirements for the degree of
**Master of Science**
**in Manufacturing Systems Engineering**

**University Supervisors**
Dr. A.E. (Alp) Akçay
Dr. W.L (Willem) Jaarsveld

**Company Supervisors**
Dr. Jan Korst
Dr. Mauro Barbieri

November 2018

# Abstract

The purpose of this research is to develop a decision support model that can help to make smarter decisions about whether to perform periodic maintenance inspections for medical devices. The decision support model uses a data-driven algorithm to predict the outcome of maintenance inspections. Together with the consideration of several operating conditions, such as costs, safety and regulation measures, this information is used to determine which course of action, i.e. conducting or skipping the planned maintenance inspection, brings most value. The decision support model is based on a binary classification algorithm using random forests. This research project is performed with the CRISP-DM methodology in mind. The approach for the construction of the decision support model is designed to be generic enough to be usable for different maintenance inspections, but this research presents a case study for a specific maintenance inspection in the context of Philips using this approach. The approach spans all the steps in the CRISP-DM process, including data understanding, data preparation, modeling and evaluation. The results show that the model has good predictive power, but the value it brings is strongly dependent on the data and operating conditions. Finally, three scenarios for the operating conditions and constraints are considered: a strict, medium and lenient scenario. Under some assumptions, the results suggest that approximately 30%, 60% and 83% of the maintenance activities in the case study can be reduced respectively.

# Executive Summary

Philips is a large organization active in the healthcare sector. Among many other medical devices, they produce interventional X-ray (iXR) devices for customers all over the world. In addition to sales, they also provide maintenance services. This research focusses on the planned maintenance services for Allura iXR devices.

### Research Goal

The goal of this research project is to improve the cost-effectiveness of preventive maintenance operations for IXR devices by exploring ways to transition part of the planned maintenance activities from a periodic-based maintenance approach to a predictive-based maintenance approach. In particular, this project has the following research goals.

- Development of a data-driven decision support model applicable to certain planned maintenance activities (performance tests) to decide whether maintenance activities should be performed or skipped.
- The model should be generic in the sense that it must be applicable to more than just one specific performance test.
- The proposed decision support model is applied to a single planned maintenance activity (the monitor performance test) to evaluate the performance and the value of the model.

### Research methodology

The research methodology that is used for this research is the CRISP-DM (cross-industry standard process for datamining). The CRISP-DM methodology is widely used methodology and it is particularly useful for this research project, because it considers a range of relevant phases, including initial problem understanding, data collection, modeling, and evaluation.

### Problem definition

The primary goal of this research project is to build a decision support model for Philips that can determine if a planned performance test should be performed at a specific point in time for a single (part of a) device. The current rationale behind conducting performance tests is to identify systems that are not operating within the allowed, predefined specifications with respect to a system aspect. In other words, the performance test is an inspection of the system. Systems that operate outside their allowed, predefined specifications are referred to as faulty systems.

Ideally, a performance test should only be performed in situations where the system is faulty, because the performance test can identify the problem and trigger appropriate follow-up actions. In cases where the system is operating correctly, the performance test is ideally skipped, since conducting the performance test would not lead to the identification of a faulty system and could thus be considered unnecessary. In total, four scenarios exist if we consider the option to skip a performance test.

The outcome of a performance test is an element of $Y = \{+, -\}$, where $+$ represents a *failed* performance tests and – represents a *passed* performance test. The real outcome of the test is denoted with $y$ and the predicted outcome is denoted with $\hat{y} \in Y$. Figure 1 summarizes the four scenarios and their associated terminology in the field of binary classification.

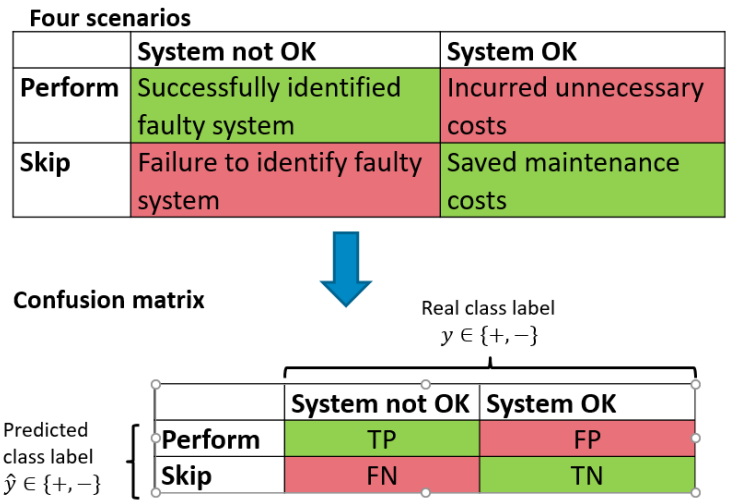



*Figure 1 - Representation of the problem along with the binary classification interpretation*

Given that extensive data per device is available, there is an opportunity to build an intelligent model that can predict the outcome of an inspection. Then, the prediction can be used to make smarter decisions about (not) performing maintenance operations. This problem can be interpreted as a binary classification problem.

## Data Collection

In order to build a data-driven decision support model, an overview of the available data is required. Over the years, Philips has collected a large amount of data from a variety of medical devices and business processes. For research purposes, Philips has aggregated a large part of the data in a single database. An initial data exploration has been done to gather understanding of the available data and the relevance to the research goals. This initial exploration led to the identification of three groups of data, each containing different kinds of information that might be useful in addressing the research goals. The groups are:

- Device and maintenance data
- System log data
- Performance test data

In addition, data relating the to a specific performance test is extracted and processed such that the proposed decision support model be applied to a performance test to gain understanding of its performance and value. For this process, a java application has been written that preprocesses the data such that it can be used for modeling. During this process, many decisions are taken regarding data preparation.

## Model

The proposed decision support model is separated into two parts. The first part consists of a binary classification algorithm. Its purpose is to predict the system's condition on the aspect that the performance test evaluates. It takes a feature vector $x$ and generates a prediction in the form of a score $s(x)$, representing the probability estimate of the model that the system is faulty. The second part of the decision support model is a decision rule considering the probability estimate of the classification model

as well as the operational context, defined by operating conditions and constraints, to generate a suggested course of action. The concept of the decision support model is shown in Figure 2.

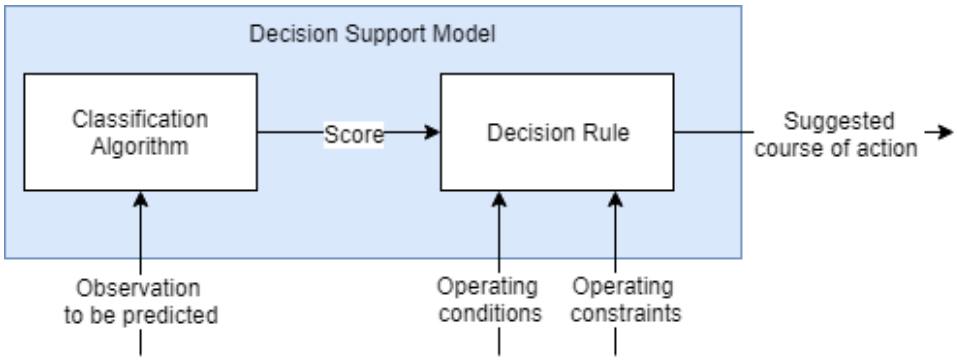



*Figure 2 - Abstract overview of the decision support model.*

The decision rule is responsible for converting the score $s(x)$ from the classification algorithm into a suggested course of action. Essentially the decision rule uses the operating conditions and operating constraints to determine a certain threshold $T$. Then $\hat{y}$ is assigned as follows:

$$\hat{y} = \begin{cases} + \ if \ s(x) \geq T \\ - \ if \ \ s(x) < T \end{cases}$$

Alternatively, the use $T$ can be described as a trade-off variable that influences the way $\hat{y}$ is classified such that the total decision support model behaves desirable with respect to the operating conditions and constraints that characterize the operational context the model is to be applied.

**Results**

To gain insight in how well the proposed decision support model functions, a case study has been performed on a single performance test: the *monitor performance* test. During the case study, many different implementations of the proposed decision model have been tested. Implementations differ on a variety of modeling aspects including hyperparameters, feature selection, data preparation decisions and the binary classification algorithm used. Among these implementations, a basic model is defined that is used to benchmark all the different implementations and it represents the general performance of the proposed decision model for the monitor performance test. The results of the basic model are expressed in ROC space shown in Figure 3.

The cost slope is included as well to illustrate the full purpose of the model, but note that this slope is based on operating conditions, which are not known in advance. Therefore, a realistic, but arbitrary cost slope is used.

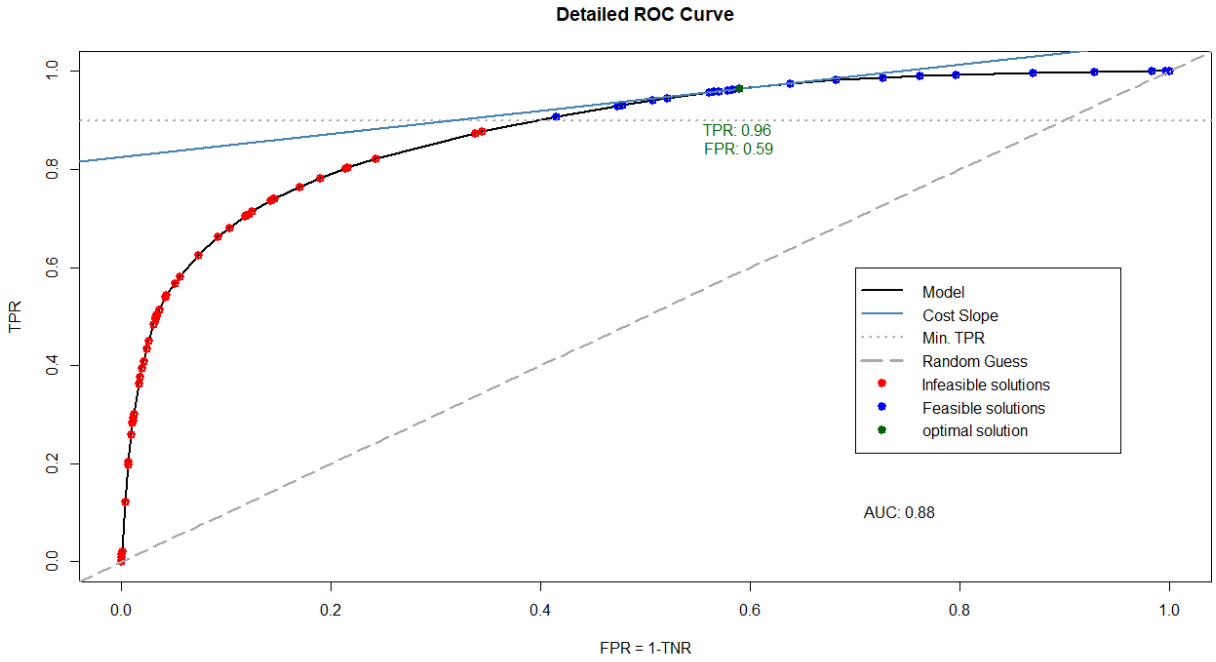



*Figure 3 - Performance of the basic model expressed in ROC space.*

Finally, to get insight into the value of the defined basic model, three scenarios are examined in more detail. The three scenarios represent a strict, medium and lenient case with respect to how much misidentified faulty systems is acceptable. Each scenario is a specific point on the ROC curve resulting from the basic model. For each scenario, the expected conducted and saved performance tests are determined, as well as the number of misidentified faulty systems. The results are summarized in Table 1.

*Table 1 - Practical performance example as a percentage of N (number of yearly performance tests)*

| Scenario | $TPR_{min}$ | FPR | Tests performed | Test saved | FN (misidentified faulty systems |
|---|---|---|---|---|---|
| **Strict** | 0.98 | 0.68 | 70.4 % | 29,6 % | 0.16 % |
| **Medium** | 0.85 | 0.29 | 33.5 % | 66.5 % | 1.2 % |
| **Lenient** | 0.70 | 0.12 | 16.6 % | 83,4 % | 2.4 % |

As can be observed from Table X, even for the strict scenario approximately 30% of the performance tests could be saved. This suggests that even in the strict scenario significant savings can be obtained. In any case, the results in table X show that the value of the model strongly depends on how the operating conditions and constraints are chosen.

# Table of Contents

# 1. Introduction

The goal of this research is to acquire knowledge to transition from a periodic maintenance approach to maintenance approach characterized by condition-based and predictive maintenance concepts in the field of healthcare technology. Maintenance is an important aspect for medical devices. Primarily because most healthcare devices must meet strict safety regulations, which means that reliable and safe use of medical equipment must be ensured trough additional maintenances actions. In addition, medical devices are important and expensive assets in the hospital environment, whose downtime is costly. Appropriate maintenance strategies can be implemented to minimize downtime. These reasons explain why it is common that maintenance costs are a major part of the total lifetime costs associated with the device. Hence, it is in the interest of both the supplier and the customer to deploy effective maintenance strategies that reduce downtime, minimize cost and improve the reliability of the system. This research considers one type of medical device in particular: interventional X-ray (IXR) devices. Interventional X-ray devices are medical systems that are used in the field of image-guided therapy. Image-guided therapy can be explained as the use of medical imaging to plan, perform and evaluate medical interventions. IXR devices are used to obtain live imaging (based on X-rays) of soft tissues during interventions.

## 1.1. Company Introduction

Philips has been founded back in 1891 by Gerard Philips and his father Frederik in Eindhoven, the Netherlands. Philips started with production of carbon-filament lamps and quickly became one of the largest producers in Europe. As the business was growing, Philips gained many international customers. In 1912, Philips became a limited company, with publicly traded shares, listed on the Amsterdam Stock Exchange. In 1914, Philips established a research laboratory (NatLab) to study physical and chemical phenomena and stimulate product innovation. Philips has diversified over the years as well. As early as in the 1920's Philips already manufactured X-Ray devices for medical diagnostics.

Philips is largely known for both their strong presence in the lighting and healthcare industry. Since 2016, Philips has split off their lighting business, which has recently adopted the new name Signify N.V. Philips still own a large share of their former lighting business but has announced in their 2016 annual report that they will continue to reduce their stake to zero. This move was part of their strategic goal to become a focused leader in health technology. Since then, they focus on two divisions in particular: consumer healthcare and (professional) healthcare systems.

The headquarters of Philips are nowadays located in Amsterdam. However, the largest part of their research division is still located in Eindhoven on the High Tech Campus, the city of their roots. In 2016, Philips employed approximately 114,000 FTE and their sales totaled EUR 24.5 billion of which more than EUR 17 billion was part of their HealthTech Portfolio.

## 1.2. Maintenance operations of Philips

Over the years, many type of maintenance approaches have been developed for all kind of industries. All these maintenance strategies can be grouped according to a high-level categorization of maintenance approaches. This section outlines this high-level categorization and considers the position of maintenance activities for Philips iXR devices in relation to this high-level categorization to put the current maintenance approach into perspective.

The first category is the *run-to-failure* or purely *corrective* maintenance policy (Susto, Schirru, Pampuri, McLoone, & Beghi, 2015). This is the most conservative and unsophisticated way of dealing with maintenance. With this approach, maintenance is performed only when the system needs maintenance, because it is considered to have failed. Because of its simplicity, this maintenance approach is widely implemented in many industries. However, researchers in the field of maintenance agree that it is also the least effective approach for most, if not all, cases. Instead, a slightly more sophisticated maintenance strategy is also widely adopted: *preventive* maintenance. This approach to maintenance is often referred to as *scheduled* maintenance, *time-based* maintenance or *periodic* maintenance (PM). With these approaches, maintenance is performed according to a planned schedule, which is often based on time intervals or usage patterns. Typically, this maintenance approach prevents a large proportion of the failures that would have occurred with a run-to-failure strategy, but arguably, this approach is still not ideal as it could involve unnecessary maintenance. In practice, this approach usually still includes corrective actions from failures, because preventing all breakdowns is rarely feasible or cost-effective. Finally, the most sophisticated category of maintenance activities is *predictive* maintenance or *condition-based* maintenance (CMB). The concept of these maintenance approaches is about evaluating the status or condition of a device to accurately predict impending failures and use that information to determine if maintenance is required. The process of predicting these pending failures varies widely among the different approaches and can be based on several pieces of information such as historical data, performance inspections and engineering expertise. Ideally, this approach should result in minimal maintenance operations, while preventing all associated failures. However, in practice it often proves difficult to accurately evaluate the system condition or predict impending failures.

Considering the current maintenance operations of Philips, it is evident that their maintenance operations largely fall under the category of preventive maintenance. This is a typical situation for highly regulated environments, because quality assurance procedures are frequently subject to safety and regulatory requirements (Hashemian & Bean, 2011). However, this does not mean these systems cannot benefit from CBM. In a survey about CBM, the concept of *reliability centered* maintenance (Prajapati, Bechtel, & Ganesan, 2012) is discussed, which enables the formulation of the maintenance strategy by selecting the right mixt of corrective, preventive and condition based maintenance. This research aims to enhance the planned maintenance operations of Philips by looking at ways to transition part of the PM activities from a preventive nature to a more prediction or condition-based nature.

## 1.3. Research Goal

The goal of this research project is to improve the cost-effectiveness of preventive maintenance operations for IXR devices by exploring ways to transition part of the PM activities from a periodic-based maintenance approach to a predictive-based maintenance approach. In particular, this research aims to develop a decision support model for Philips to periodically decide whether to perform a certain PM activity on a given device. The model aims to leverage several sources of data to tailor the output of the model to individual devices. Ideally, periodic adjustments are the result of a data-driven approach that enables Philips to determine if a certain PM activity is not adding any value, which results in the opportunity to skip the PM activity and prevent unnecessary maintenance. Since PM activities for iXR devices are rather diverse in terms of execution, purpose and data, this research primarily focusses on a single set of PM activities. These PM activities are called *image quality (IQ) performance tests* and their purpose is to ensure the quality of the image chain. This research project aims to function as a proof of

concept for a data-driven approach to reduce PM costs at Philips and pave the way for other, similar data-driven PM initiatives. During this research project, the following research goals are pursued:

1. Develop a decision support model that determines if a scheduled PM activity should be performed during the next planned maintenance visit. The model has the following characteristics:
    a. The model uses a variety of input variables such as usage patterns, system conditions, previous test results and demographic variables to determine its output.
    b. The model behavior should be responsive to a set of business input parameters that represent the trade-off between the value of performing the PM activity versus skipping the activity, as well as several operating conditions.
2. The approach towards building the decision support model must be generic enough to be applicable for other, similar PM activities within Philips. However, the proposed decision support model in this research is applied to a single PM activity.
3. Evaluate the performance of the model to explore what the impact it would have if it were to be implemented.

## 1.4. Scope

The goal of this research is to develop a model that can improve the cost-effectiveness of the planned maintenance operations for medical devices. However, PM is a general term for a healthcare organization and therefore the scope of this research project is defined in terms of two aspects: medical equipment and the type of maintenance.

### 1.4.1. Devices

Since Philips produces a wide variety of medical equipment, we define the scope of this project in terms of medical equipment. As mentioned throughout this research, the main focus is on preventive maintenance of interventional X-Ray (iXR) devices. Over the years, Philips has developed several generations of these devices. For this research, the scope is limited to all the devices that are part of the *Allura* generation. An example of an Allura iXR device shown in Figure 4. This generation of devices currently has the largest installed base. Philips has recently introduced a newer generation of devices referred to as *Azurion*. The purposes of the devices are the same, but the newer devices have additional and more modern features.

At the time of the start of this research, it was not as interesting to look at as *Azurion* systems, because the installed base was still relatively small. In addition, not much historical data is available due to the recent market introduction. Therefore, in the remainder of this research the term IXR devices refers to only Allura IXR devices.

Please note that not all IXR devices are included in the scope of this research. Some devices are not connected to Philips' databases or are maintained by third parties. As such, only devices with service contracts and a relatively complete set of data are considered.

*Figure 4 - A picture of an Allura Interventional X-ray device.*

### 1.4.2. Diversity in PM Activities

Even if only PM activities from the IXR devices are considered, PM tasks still differ in many ways such as purpose, data availability, related subsystems and frequency of execution. Based on an orientation of the PM tasks, they are grouped in four categories, shown in Table 2.

*Table 2 - Four categories of PM tasks for the Allura IXR devices.*

| Type of PM Task | Time fraction |
|---|---|
| **Preparation, Administration & Support** | 26% |
| **Adjustments & Calibrations (Direct adjustment without prior testing)** | 34% |
| **Performance Tests & Safety Checks (No Adjustment, just tests/inspections)** | 32% |
| **Conditioning & Cleaning (Affects system condition)** | 8% |

Since the purpose of this research is to develop a decision support model to improve PM operations, it is clear that a focus on one category of PM activities would be the preferable approach, because a narrow focus would allow for a deeper and more extensive analysis. Particularly, the group of calibrations as well as the group of performance test seemed interesting. Both categories were responsible for a large proportion of the total spent PM time. In addition, both also had a large set of related data. However, after a brief exploration of the data, it turned out that in general calibration data was not as useful as that of the performance tests. Hence, the decision was made to focus on the category of performance tests. Even then, there were many different performance tests with differences in data availability, quality and

other implications. In the end, a group of performance test was chosen as the basis for this research. The tests in this group are called *Image Quality (IQ) Performance Tests* and their purpose is to ensure that the whole chain from the generation of X-rays to the final image on the screen meet all predefined quality standards. In particular, this research is focused one single IQ test: *The Monitor Performance Test*. However, the proposed decision support model is designed to be applicable for other IQ tests as well. The monitor performance test is chosen, because a good amount of data is available and this test seemed to have the fewest underlying interfering factors (such as safety, data and maintenance implications). For example, for many other tests, one would have to account for actions that would influence the condition of the device.

## 1.5. Rationale of the Decision Support Model

The primary goal of this research project is to build a decision support model for Philips that can determine if a planned performance test should be performed at a specific point in time for a single (part of a) device. The current rationale behind conducting performance tests is to identify systems that are not operating within the allowed, predefined specifications with respect to a system aspect. In other words, the performance test is an inspection of the system. For the sake of clarity, systems that operate outside their predefined allowed specifications are referred to as *faulty* systems.

Ideally, a performance test should only be performed in situations where the system is faulty, because the performance test can identify the problem and trigger appropriate follow-up actions. In cases where the system is operating correctly, the performance test is ideally skipped, since conducting the performance test would not lead to the identification of a faulty system and could thus be considered unnecessary. In total, four scenarios exist if we consider the option to skip a performance test. Table 3 presents the four scenarios.

*Table 3 - Scenario matrix where green represents an advantageous scenario and orange represents a disadvantageous scenario.*

| | **System not operating correctly** | **System operating correctly** |
|---|---|---|
| **Perform Test** | Successfully identified faulty system | Incurred unnecessary costs |
| **Skip Test** | Failure to identify faulty system | Saved maintenance costs |

In practice, it is not known beforehand if the system is faulty or not. In fact, that is the primary reason for conducting the performance test in the first place. Therefore, the performance tests were introduced and performed according to a fixed schedule to timely identify faulty systems. However, with the availability of extensive data, an opportunity arises to build an intelligent model that can predict the outcome of an inspection. Then this information can be used to decide what course of action should be taken such that we end up in the scenarios in Table 3 marked with a green color.

The situation described above can be described as a binary classification problem. Hence, the decision support model is based on binary classification model that predicts the outcome of an inspection. To make the model appropriate and relevant for Philips, an additional component is added. This component is a decision rule, which uses the classification of the binary classifier along with certain operating conditions and requirements, such that decision making is optimized with respect to the operational context in which the model is applied. The operational context is characterized in terms two components. The first component are costs associated with each type of misclassification (represented by the red cells in Table 3). The second component is the class skew, which essentially reflects the portion of systems that are

(not) operating correctly based on historical data. In addition, two operational constrains are introduced to adequately limit potential safety concerns that may arise when using the decision support model. The decision support model is described in detail in Chapter 5.

## 1.6. Thesis Outline

Chapter 2 discusses the methodology that is used for this research project. The remainder of this report is structured according to the discussed methodology, starting with Chapter 3 that discusses the current maintenance operations of Philips. Subsequently, Chapter 4 discusses the data understanding and data preparation phase. Next, Chapter 5 discusses the development of the generic decision support model for planned maintenance activities in the healthcare environment. Chapter 6 discusses a case study where the model is applied to one of the planned maintenance activities. Finally, Chapter 7 and 8 discuss the conclusions and future work respectively.

# 2. Research Methodology

This chapter discusses the research methodology that is used for this project. All the steps that are taken in this research project are based on the CRISP-DM methodology. This methodology is one of the most-widely used methodologies for datamining and data analytics. The framework has been developed in 1999 by a consortium of leading organizations in the field of data mining at the time (Pete et al., 2004), (Wirth & Hipp, 2000). The project was partly funded by the European Commission under the ESPRIT program.

Sometimes the term data mining is misunderstood, because the name suggests that it is purely about the extraction of data. However, as the field of data mining has exploded over the years, the term nowadays encompasses a wider range of data-related activities. In a similar way, the book Data Mining: Concepts and techniques (Han, Kamber, & Pei, 2011) argues that the term data mining is a misnomer and that it can refer to several things, including data analysis. This observation is important, because at a first glance one might argue that this research project is not just about datamining, but about building a decision support model for maintenance operations within Philips. Following that reasoning, data mining would be a single step, rather than the whole process. Hence, it could be argued that a DM-methodology would not be appropriate for this research. However, as will become clear during the discussion of the concept of the CRISP-DM methodology, it does contain the required steps cover the whole process from understanding the business needs all the way to the deployment of a model.

## 2.1. The CRISP-DM Methodology

The CRISP-DM methodology is a widely used, application-neutral methodology for data analysis and is an abbreviation for *Cross Industry Standard Process for Data Mining*. This paragraph briefly describes the concept of the CRISP-DM methodology. For a more detailed overview the reader is referred to the step-by-step user guide (Pete et al., 2004), written by members of the original consortium that was responsible for developing this methodology.

In the step-by-step user guide, the authors capture the essence of the model with the following description:

> *"The CRISP-DM methodology is described in terms of a hierarchal process model, comprising four levels of abstraction (from general to specific): phases, generic tasks, specialized tasks and process instances (see Figure 5)".*

The idea behind the hierarchal process model is that each level of abstraction becomes less generic and more specific to the project that it is applied to. As can be observed from Figure 5, the two upper levels are generic, and the two lower levels are specific to the project. The reason that the top levels are generic, is because this methodology was designed to be usable for any data-mining project in any industry. As such, it is up to the user of this methodology to give meaning to the lower abstract levels depending on their project characteristics. The most generic abstraction level consists of phases, which are present in most data-mining projects. The CRISP-DM methodology uses the CRISP-DM reference model to characterize the relation between the phases, shown in Figure 6. Each phase consists of a set of generic tasks that belong to the second level. These tasks are generic in the sense that they are defined independently from a project implementation. For example, the step *data cleaning* can involve many different things depending on the type of project. In some cases, it might even be excluded if it is does not add value to the project. This is where the third abstract level comes in, which is organized as specialized tasks that result from mapping the CRISP Process Model to the datamining project at hand.

The fourth and lowest abstraction level is referred to as process instances and these contain records of actions, decisions and results of actual data mining efforts.

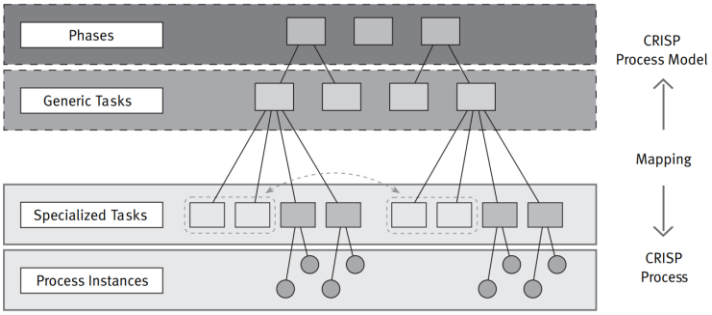



*Figure 5 - Four level breakdown of the CRISP-DM methodology*

Ideally, the phases capture an optimal sequence of steps in the datamining project. However, the authors acknowledge that in practice many of the phases can be addressed in a different and iterative sequence, which in some cases is even a necessity. In addition, it is important to note that the authors describe a reference model, suggesting that it is the responsibility of the researcher to correctly map relevant steps in the project to the reference model. In a way, the reference model is a sophisticated checklist for data-mining users that can be used to identify and define the relevant steps in their project. Therefore, the way the researchers map the CRISP Process Model to the project essentially defines the research approach, because the result of the mapping defines which steps are to be taken and how.

The final paragraph in this chapter is dedicated to describing the application of this CRISP-DM methodology specifically to this research project. That is, the paragraph presents a table of specialized tasks along with motivation for each task.

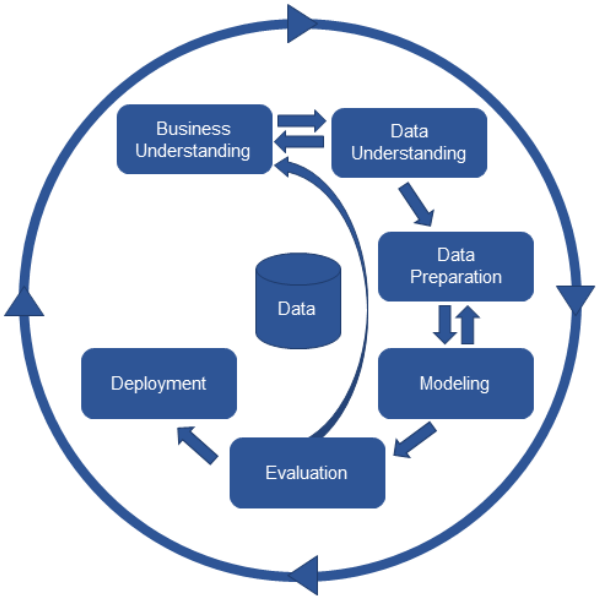



*Figure 6 - Phases of the CRISP-DM reference model.*

## 2.2. Project Approach

In this paragraph, the project approach is refined by mapping the project to the generic phases in the reference model. The CRISP DM methodology provides a generic list of tasks for each phase, shown in Figure 7.

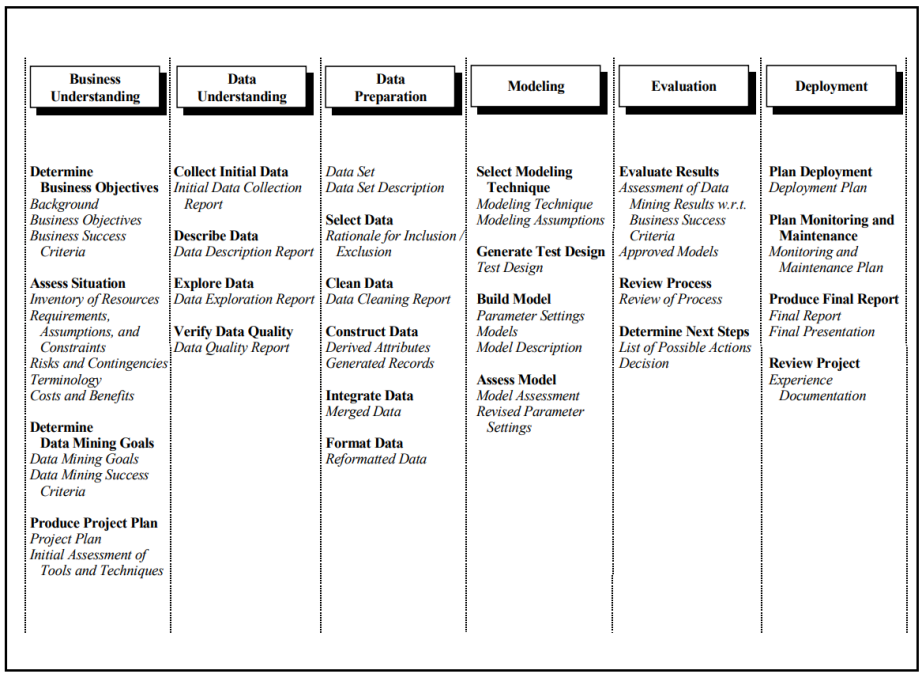



*Figure 7 - Generic tasks(bold) and outputs (italic) of the Crisp-DM reference model.*

This generic list is used as a guideline to define the approach for this project. The authors encourage to leave out irrelevant tasks and include additional tasks if they provide value to the project. The mapping process results in a basic approach for the project, but note that mapping is done continuously throughout the project, because it is difficult to map all the activities to each phase before any of the work is performed. The result is summarized in Table 4. Section 2.2.1 trough Section 2.2.5 briefly discuss the defined actions for each of the phases.

*Table 4 - Summary of project approach.*

| **Business Understanding** | **Data Understanding** | **Data Preparation** | **Modeling** | **Evaluation** |
|---|---|---|---|---|
| **Chapter 1 & 3** | **Chapter 4** | **Chapter 4** | **Chapter 5** | **Chapter 6, 7, 8** |
| Determine Business Objectives and project goals<br><br>Asses maintenance operations | Explore available Database<br><br>Interpret the Data | Data selection<br><br>Data aggregation<br><br>Feature selection<br><br>Data cleaning<br><br>Data output | Select modeling Technique<br><br>Consider performance metrics<br><br>Implement operational conditions and constraints<br><br>Implement a decision rule to transform the prediction in a suggested course of action | Evaluate results by applying model to a specific performance test<br><br>Consider importance of<br>- Hyperparameters<br>- Data cleaning decisions<br>- Feature importance<br><br>Consider alternative model: two-stage learning algorithm |

### 2.2.1. Business Understanding

During the business-understanding phase, two primary goals are pursued. The first goal is to determine the business objectives in terms of the planned maintenance strategy. This is achieved by acquiring input from the respective Business Innovation Unit (BIU), which are responsible for the high-level organization of the planned maintenance activities. Based on their vision and goals, the right project goals and criteria are defined.

The second goal is to get a thorough understanding of the current maintenance operations within Philips. It is important to assess the current situation, because that should reveal practical constraints, oddities, opportunities and other relevant information for this project. First, the general maintenance operations within Philips for IXR devices are described. After that, a closer look is taken at how the PM schedules are constructed, because that provides more information on what options there are to deviate from current maintenance schedules. Finally, we zoom in on of the specific PM activities that are used as the case study for this research project: *The Monitor Performance Test*.

### 2.2.2. Data Understanding

The goal of this phase is to get a good understanding of the available data. This process starts by exploring a large database that is primarily constructed for research purposes. It contains (processed) data from a variety of sources, including the IXR domain. The exploration provides knowledge on what data is available, what type of data it is and how it is all linked together. In addition, this phase requires considerable interaction with experts from Philips, since the knowledge about certain topics is distributed amongst different people, departments and documents.

### 2.2.3. Data Preparation

In the data preparation phase the emphasis is on preparing the data such that it is usable for modeling. This is an important, but time-consuming step where many decisions need to be taken. For this research, five tasks similar to the generic tasks from the CRISP-DM methodology are used. First, potential data identified during the data understanding phase is extracted from the database. Second, the data is aggregated. This is necessary, because data originates from different sources and usually has different formats. Aggregating the data is essential, because it allows us to identify missing and/or unreliable data. Third, the data is cleaned. During this step, reasons for including or excluding data are carefully documented, because that helps to prevent unjust interpretations and conclusions during the evaluation of the results. Next, the features are derived from the extracted data. This step aims to characterize the device by means of input variables, which can be used during the modeling stage to predict the outcome of a performance test. Finally, the data is formatted and exported such that it can be used for modeling.

It is common knowledge in the field of data science that knowing the data is essential for good modeling, Hence, an overview of the final dataset is provided along with some descriptive statistics in the data preparation phase.

### 2.2.4. Modeling

The aim of the modeling phase is to build a generic decision support model that can be used to generate a suggested course of action with respect to conducting a certain performance test during the next planned maintenance visit for a given device. During this phase, the concept of the decision support model is defined, and several important aspects are considered such as the modeling technique(s) and associated performance metrics. In addition, the operational context is considered which characterizes the

environment the model is applied to. Finally, a decision rule is discussed that uses the operational context to adjust the model behavior to be in line with the operational context.

### 2.2.5. Evaluation

In this phase, the decision support model is evaluated by applying it to one of the performance test: *the monitor Performance test*. Firstly, a basic model is defined, meaning that generic model is implemented according to a set of hyperparameters and modeling decisions. Then, experiments are conducted to see how different modeling decisions and hyperparameters affect the performance of the model. In addition, the model is tweaked to a two-stage-learning algorithm in an attempt to achieve better performance. Subsequently, the basic model is validated with an additional validation set. Finally, the performance of the model is translated to the potential business impact.

# 3. Maintenance Process Philips

This chapter provides an overview of the maintenance operations that Philips performs to maintain their IXR devices. Section 3.1 discusses the difference between preventive maintenance and corrective maintenance within the context of Philips. Finally, Section 3.2 gives more detail on the performance tests for which the decision support model is developed.

## 3.1. Preventive and Corrective Maintenance

Maintenance operations within Philips are organized by means of cases. All maintenance cases can be split into two categories: preventive maintenance and corrective maintenance (CM). Philips defines preventive maintenance in the following way

> *Preventative Maintenance is used for the service request to deliver the scheduled maintenance activities which are contractually agreed upon in advance by a warranty or service contract, or purchased by the customer. (Philips Internal Document: Service coding, GCS-GSP-CD-00003).*

Within Philips, each IXR device has a preventive maintenance schedule. System and safety experts create the PM schedule during the R&D phase of the device. The purpose of PM is to ensure that the device is operating correctly within specifications, to avoid unscheduled downtime and to meet safety and quality requirements. The PM schedule is based on a maintenance cycle of two years. Philips performs this maintenance cycle as long as customers have an active service contract with Philips. Some customers choose not to sign a service contract, which results in the fact that Philips will only perform PM during the warranty period, which is typically one year, or when customers purchase the services separately. Typical examples of PM activities are the lubrication of the bearings, calibration of the X-Ray dosage and image quality tests. In addition, it is important to consider how PM schedules are constructed, because that provides information regarding the possibilities and constraints when changing maintenance schedules. Appendix A describes the process of constructing a PM schedule.

In addition to PM, Philips defines CM in the following way:

> *Corrective Maintenance is used for the service request to fix a problem of broken equipment; the fix may be delivered remotely with or without parts, onsite with or without parts, or delivered in a bench repair center. (Philips Internal Document: Service coding, GCS-GSP-CD-00003).*

In practice, CM activities are the result of a (impending) system failure. However, there are several ways CM cases can be initiated and these are illustrated in Figure 8. A CM case is either initiated from the side of the customer or from the side of Philips. In case the customer initiates the CM case, it is most often because the customer has a complaint or notices a problem with the device. Alternatively, Philips initiates a CM case for one of the following reasons. It may be that the field service engineer notices a problem during one if his regular PM visits. If the problem is related to the actual PM activity, the FSE will try to fix this during the visit. However, if the problem seems unrelated a separate CM case is created. It may also be that a maintenance case is initiated automatically if the system on the device is able to identify a problem on its own. Finally, it can be the case that the remote monitoring team notices a problem with a device based on the system logs that are sent over daily. These system logs contain warning and error messages that are analyzed by models from the remote monitoring team to raise alerts for patterns that relate to frequent system breakdowns. Depending on several factors such as the local market, confidence of the alert and human expertise the alert may lead to a CM case.

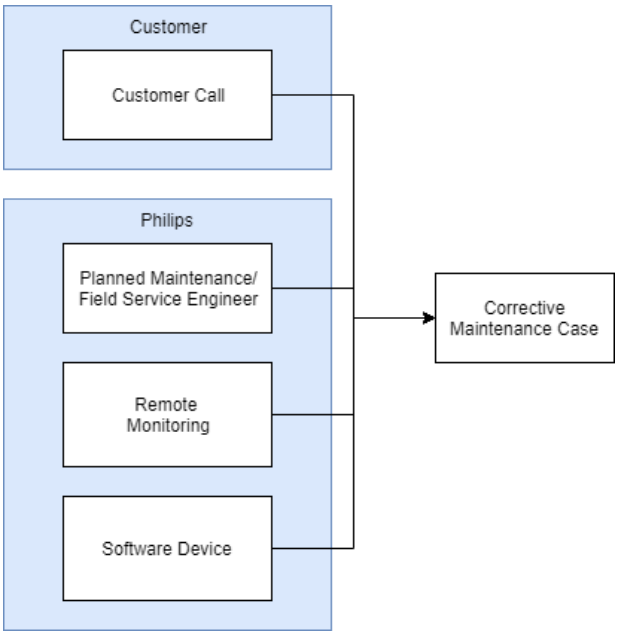



*Figure 8 - Four ways a corrective maintenance case can be initiated.*

Each maintenance case, both CM and PM, have a priority code which is used in conjunction with the service agreement of the customer to determine the urgency of response to the maintenance case. PM cases always have the lowest priority, whereas CM cases can have values ranging from one to five.

## 3.2. Performance Tests

A significant part of the planned maintenance schedule consists of PM activities that exist to verify that the overall performance of the system is in line with the predefined specifications set during the R&D stage of the device. In this research, we refer to these PM activities as *performance* tests. Each performance test is responsible for checking a specific part or subsystem of the device trough a set of quality assurance measurements that evaluate the performance under various condition and loads. Typically, these tests address parts of the system where failures are not self-evident and in the field of maintenance, this is often referred to as *hidden* failures or *silent* failures. In practice, this means that equipment users will most likely not notice when the device is starting to deviate from the designed state. To illustrate the importance of these tests, consider the scenario where a patient is exposed to a heavier X-ray dosage than intended. As is commonly known, X-rays can be harmful to the human body, especially in large amounts. To guarantee that the patient experiences minimal adverse health effects from the medical intervention, it is important to have a process in place that verifies the correct X-ray dosages are generated.

Although performance tests are part of the planned maintenance schedule, they are different from conventional maintenance activities in that they oftentimes do not influence the state or condition of a device. Instead, performance tests are primarily conducted to identify hidden failures, meaning that their purpose is more similar to that of a periodic inspection. This is an important observation, because it is implicitly assumed that periodic inspections do not influence the system condition on their own. As is mentioned in Chapter 1, the generic decision support model is tested on a specific performance test: *the Monitor Performance Test*. This performance test is designed to examine the performance of medical-grade monitors. It is vital that they are in good condition, because doctors and surgeons rely on these

monitors for diagnostics and interventional treatments. Figure 9 contains two examples of these medical-grade monitors and Appendix B describes the monitor performance test in more detail.


*Figure 9 - Two examples of medical-grade monitors for image-guided therapy.*

# 4. Data Exploration and Preparation

This chapter discusses two important phases relating to data that is used for this project: the data understanding phase and data preparation phase. The chapter starts with an overview of the initial data exploration and the categorization of the available data in three categories (Section 4.1). Then the chapter moves on to discuss the data preparation phase in which data is prepared such that it can be used during the modeling phase (Section 4.2). Finally, the chapter ends with an overview of the final dataset that will be used for modeling (Section 4.3).

## 4.1. Data Exploration

Over the years, Philips has collected a large amount of data from a variety of medical devices and business processes. For research purposes, Philips has aggregated a large part of the data in a single database. All data used for this this research is extracted from that database. The first step in understanding the data is an initial exploration to gather understanding of the available data and the relevance to the research goals. This initial exploration led to the identification of three groups of data, each containing different kinds of information that might be useful in addressing the research goals. The groups are:

- Device and maintenance data
- System log data
- Performance Test Data

Each group of data is discussed in a separate paragraph, elaborating on the source of the data, the format and why it is relevant to the research goals.

### 4.1.1. Device and Maintenance Data

Device data refers to all the data that is related to a specific IXR device. The used data fields are the equipment number, the system code and the installation date and country. These data fields are used to uniquely identify a device, as well as determine its type, age and location. In a later stage, the unique equipment number is used as an identifier to couple other sources of data to the device. Together, these data fields are used to characterize each device in the dataset. Table 5 shows an overview of the data fields.

*Table 5 - Extracted data fields to characterize a device.*

| Name of Column | Description |
|---|---|
| **equipment number** | Unique ID for an IXR device |
| **system code** | Code that refers to the specific model of this device |
| **installation date** | Time at which the device was installed at the customer |
| **country** | The country where the device is installed |

In addition, there is related maintenance data for each device, which contains records of all the maintenances activities Philips performs for that device or customer. The data specifies the kind of maintenance, time of the maintenance and the resources required for the maintenance (labor hours and replaced parts) and a short description of the maintenance. For this research, part of the maintenance data is analyzed to identify relevant part replacements for each device, which is discussed later in Section 4.2.

### 4.1.2. System Log Data

The second type of data we look at in this research is system log data. Throughout its operational lifetime, any IXR device generates a significant amount of log data. This data is called system log data and is collected and stored in system log files. Given that the device is properly connected to Philips' databases, the log files are sent to Philips daily. System log files contain detailed information about system events. For example, such an event might represent the device turning on, the X-ray arm rotating to the left or the system detecting its configuration. The data in the system log files is in a raw form. Therefore, Philips uses extract-transform-load (ETL) tools to extract relevant data and store it in databases for later use.

Since there is no clear understanding about the degradation of IXR (sub)systems, this system log data is essential for the concept of predictive or condition-based maintenance, because it potentially represents the hidden footprints that characterize system degradation. This data can be used to extract characteristics and usage information. For example, it can be used to determine the number of treated patients, the most frequently used application type, the system uptime or the configuration with respect to the type of monitors used. These extracted characteristics are used in a later stage for modeling purposes.

### 4.1.3. Performance Test Data

The third type of data we look at in this research is performance test data. It holds information about the results of the executed performance tests during a planned maintenance visit. Technically, this data is similar to the system log data, because the quality assurance measurements are stored in the system log files. However, it is different in the sense that performance test data is only generated during maintenance activities and in the presence of a field service engineer. In addition, this data is considered as a different group, because part of this data is also stored in a separated database using a performance assurance tool. Field service engineers use this tool to store the vital parts of the quality assurance measurements in a different database.

The performance test data that is used for this research is stored in the Vertica database and each row of data represents a specific subtest. To understand what each field means, it is useful to first consider how PM visits, performance tests and subtests are related. In practice, a PM visit occurs on a specific date for a specific device. During each PM visit, zero or more performance tests may be performed. Then, for each performance test the outcome is determined based on the results of all its constituent subtests. The subtests contain the actual quality performance measurements that represent the condition of the system.

Each row of data contains information about one subtest. Each row of data consists of several fields and we distinguish between two types of fields: information fields and relation fields. Data in information fields represent outcomes and characteristics of the conducted performance tests. Data in relation fields is used to correctly group subtests that belong together. For example, the following fields are used to group data together correctly: *date and time*, *test name* and *equipment number*. Table 6 shows all the fields that are extracted for this research along with an example and short description. The gray rows represent the data fields that are relation fields. These relations are important, because at a later stage, knowing what subtests belong to the same performance test is a necessity in the analysis.

*Table 6 - Extracted data fields for performance tests. Gray rows represent relation fields and white rows represent information fields*

| Name of Column | Example value | Description |
|---|---|---|
| **equipment number** | 123456789 | Unique ID for an IXR device |
| **result** | passed | Result of the performance test, either *passed* or *failed* |
| **date and time** | 2017-08-29 12:56:22.0 | Timestamp of the performance test |
| **test name** | Monitor Performance Test | Name of the performance test |
| **measurement name** | ExamFrontal Contrast Upper | Name of the subtest (or quality assurance measurement) |
| **measurement Result** | passed | The result of the specific subtest, either *passed* or *failed* |
| **measurement Value** | 0.75 | The numerical value measured during the subtest |
| **warning lower value** | 0.50 | The lower value for the warning range of the subtest |
| **warning upper value** | 0.90 | The upper value for the warning range of the subtest |
| **error lower value** | 0.45 | The lower value for the error range of the subtest |
| **error upper value** | 0.95 | The upper value for the error range of the subtest |

## 4.2. Data Preparation

Section 4.1 discusses the exploration of available data within Philips. In this section, several data preparation steps are discussed to transform the raw available data into data that can be used for modeling in Chapter 5. The following data preparation steps are taken:

- **Data extraction**: Process of extracting data from databases.
- **Data merging**: Merge data from a variety of tables and formats together.
- **Feature selection**: Extract and define features or characteristics for a device or performance test.
- **Data cleaning**: clean the data by excluding incomplete, inaccurate or unreliable data.
- **Data output**: output the data in such a format that it can be used for modeling purposes.

Figure 10 shows the general process of these five data activities. Ideally, these are performed in sequence, but in practice, this process is iterative because of the introduction of new data and insights at several moments throughout the project. Each of these steps is briefly described in a dedicated paragraph.

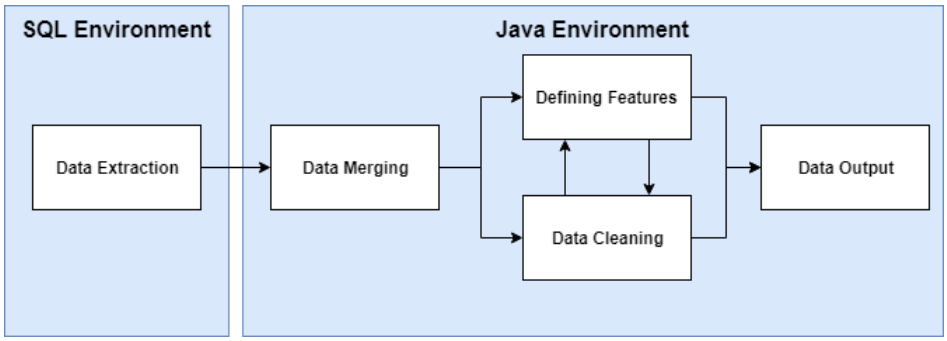



*Figure 10 - Data Preparation Process.*

### 4.2.1. Data Extraction

For this research, data is extracted trough SQL queries from the Vertica database. The specific tables that are referenced for this research are documented in Appendix C. The queries have been written in such a way to only include data that is deemed relevant. That is, only data for IXR devices is included, duplicate rows of data are excluded, and rows containing null values for important fields are excluded. In some cases, queries had to be extended to retrieve data from multiple tables to ensure that each extracted dataset contained identification fields such as *EquipmentNumber*, referring to the unique identifier of an IXR device. This necessary to successfully link the datasets in a later stage. All the extracted data is stored in dedicated files for later use. Alternatively, an active connection between the database and the modeling environment could have been set up, such that the most up-to-date data could be queried in real-time while building the model. The decision to store the extracted data in dedicated local files was a deliberate decision and is preferred because of the following reasons. First, an active connection seemed unnecessary, because the relevant data is historical data and is not supposed to change. Furthermore, implementing an active connection could result in querying data more frequently than required, potentially putting more strain on the querying servers of Philips than necessary. In addition, it guarantees that during the remainder of the data preparation stage the same dataset is used. This is beneficial, because it makes the rest of the data preparation process more transparent and robust, because no new data is flowing in that could cause unexpected behavior or results.

### 4.2.2. Data Aggregation

The next step in the data preparation process is to link all the data together. This step is important, because it makes it possible to derive attributes that are based on different sources of data. Furthermore, it is an essential step for data cleaning, because linking datasets will reveal gaps and inconsistencies in the dataset.

For the data aggregation process, Java code has been written. The code can read all the extracted data files from the data extraction step and link them together such that they can be used for data analysis. In a later stage, this Java code was extended into a small application that also included logic for feature selection, data cleaning and data output. These topics are discussed in the remaining sections 4.2.3 through Section 4.2.5.

Before any of the logic is written, it is important to consider how the data is linked. This knowledge is primarily obtained during the data understanding phase a result of the data exploration. In addition, many conversations with experts within Philips provided additional understanding. In the end, the data is aggregated according to the simplified UML-Class diagram shown in Figure 11. Note that the UML class diagram shows the structure that is used to group all the data extracted data together for processing, but it does not represent the database structure. Each block represents a data entity. Here, we define four data entities that we use to store relevant data that we wish to use. Note that the data entities are linked with compositional relations, indicating that they are dependent, which is important for data cleaning.

If we consider the *Monitor Performance Test*, which is central to this investigation, the diagram shows that each device has a set of *Monitors* and that a *Performance Test* is performed for each of those monitors. In turn, each *Performance Test* consists of a set of *Subtests*. Figure 11 represents only a part of the code structure, but provides the framework for the data merging process. Given this framework, it is relatively easy to link other data to the appropriate entity.

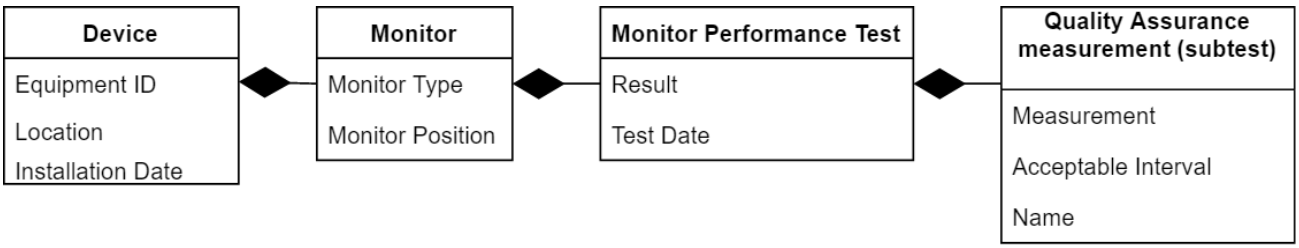



*Figure 11 - Simplified UML Class diagram used for data merging.*

### 4.2.3. Defining Attributes and Features

In the context of machine learning or classification, features are essentially a set of input variables that represent a characteristic or property of the (abstract) entity being observed. The goal of this step is to define features with good potential for predictive power. However, the predictive power of a feature can only be determined in relation to a specific classification problem and dataset. Since the decision model that is proposed in this research is designed to be generic, i.e. applicable for multiple performance tests, the value of the features cannot be defined explicitly at this stage. Consequently, the outcome of this step is a list of features that may or may not be used, depending on their predictive power in relation to the specific classification problem. Arguably, this is one of the most important steps in the data preparation phase, because it directly affects model performance. This paragraph only discusses the most relevant decisions that have been made regarding the attributes and features of the final dataset.

Selecting the right data for features can be a difficult process, because it is difficult to know upfront what data could lead to futures with good predictive power. In this project, features have been primarily selected and defined based on the available data that was discovered during the data understanding phase. Together with suggestions from system experts and data scientists, a list of potentially valuable features is defined. Considering certain data quality characteristics, this list was implemented by extracting and merging the appropriate data as described respectively in sections 4.2.1 and 4.2.2. Table 7 shows an overview of three categories of features along with a few examples.

*Table 7 - Feature types*

| **Feature Category** | **Features** |
|---|---|
| Previous Measurements and Results | • Quality assurance measurements (subtests)<br>• Relative deviation from allowed operating range<br>• Test outcome |
| System characteristics | • SystemCode<br>• Monitor Type<br>• Location (country)<br>• Age of the system |
| Usage parameters | • System-on time<br>• Number of patients treated<br>• Number of startups<br>• Most used application |

It is worthwhile to mention that some of the features are static, meaning that they will be the same for a set of related cases. An example of such a feature is the *location* of the system. However, when defining

other features, such as usage features, the time of the performance test plays an important role. It is essential that no future data with respect to the observation is included, because that data holds information about events after the observation in question.

### 4.2.4. Data Cleaning

Data cleaning is an important part of any data related project. Old or inaccurate data is likely to have an impact on the outcome of the project. Prior steps in the data preparation already revealed that this is the case with Philips data too. For this project, the data cleaning process is automated in the same Java application that is responsible for reading and aggregating the data as well as for feature selection.

There are several ways to deal with missing and incomplete data (Das, Datta, & Chaudhuri, 2018) and (Garcia-Lacencina, Sanch-Gomez, & Figueiras-Vidal, 2010). Both articles mention the concepts of *marginalization* and *imputation*. The marginalization method excludes data points with unobserved features from the dataset. Alternatively, imputation attempts to fill in the missing features by making reasonable estimates based on the values observed for the corresponding features over the rest of the dataset. There are arguments to be made for each method, but in this research, marginalization is used. Marginalization can be a good method if data is characterized by *missingness at random* (MAR), which essentially means that missing data is random and depends neither on the observed and unobserved data. Consequently, in the case of MAR, removing data from the dataset will not affect the shape of the dataset and if the dataset remains large enough, good models can be constructed. However, it is unlikely that this dataset is characterized by MAR as the presence of data in the database is likely dependent on several factors. (Das et al., 2018) indicate that when the rate of missingness is low (1%-5%), incomplete data instances can be safely skipped, and the model could still be used on all (complete) samples. They refer to this as *complete case analysis*. However, as will become clear later in this section, our data cleaning results in the exclusion of a far larger proportion of data. Given this concern, it is not justifiable to claim that a model based on such a dataset could be used for all future performance tests, because future performance tests may suffer from missingness too. Therefore, using only a subset of the available data implicitly limits the scope to which the implemented model may be applied. The authors refer to this method as *the available case analysis.* In this research, imputation is considered undesirable, because data instances that have missing features typically have a considerable number of missing features. One observed reason for this is that when a medical device is not present in a data table, all features derived from that data table are missing. Applying imputation on such data instances would lead to a very random sample in the sense that a large part of the data sample is random, instead of a real observation.

The data cleaning process works by initially marking all data entities; *Devices, Monitor Positions, Performance Tests and Subtests*, (Figure 11) as usable. Subsequently, all these data entities undergo a set of data checks. If an entity does not pass all the checks, it is marked as unusable and the reason for marking a specific entity unusable is recorded. It is important to note that marking an entity as unusable, could lead to other entities being marked as unusable too if they dependent on the rejected entity. For example, if a medical device that is excluded because its installation date and location are unknown, all the related performance tests are also marked as unusable. After all the data checks have been performed, only the entities that are still marked as usable are included in the final dataset.

Data cleaning can be somewhat of a subjective task, because it is not always clear what data should be included or excluded. For that reason, some of the data cleaning decisions are implemented in the code with variables such that they can be changed if new insights and interpretations arise. Other decisions are

fixed, since it is assumed that the decision will not change anymore. Table 8 and Table 9 summarize the fixed and modifiable decisions that are used in the data cleaning process respectively. Appendix D shows an example of the automated (low level) data cleaning report that the code generates. It shows what data has been excluded and for what reason.

*Table 8 - Fixed data cleaning decisions*

| Data cleaning decision | Explanation |
|---|---|
| Exclude the performance test if any of the constituent subtests is marked as unusable | This decision makes sure that a performance test is excluded from the dataset if any of its constituent subtests is marked as unusable. This is necessary, because the performance test is incomplete and therefore its outcome cannot be defined on the same condition as other performance tests. |
| If a data entity is marked as unusable, all its data entities that depend on that entity are marked as unusable too. | This decision excludes all data that falls under the entity that is considered unusable. For example, if a medical device is excluded from the dataset, all tests related to that device are excluded as well, because one or more features could not be determined for those tests. |
| Filter data that is in the wrong format | Some lines of data contain wrongly formatted data, which could not be interpreted correctly. For example, some fields contain a string where there should be a numerical value. |

*Table 9 - Modifiable data cleaning decisions.*

| Data cleaning decision | Default decision | Explanation |
|---|---|---|
| Exclude measurements that have a value of zero | True | These lines of data are filtered, because it is assumed that these values cannot be zero and thus are inaccurate. A closer inspection also revealed that if this is the case, usually all other subtests that belong to the same performance test are zero as well, which indicates something went wrong in the data collection process. |
| Exclude the first performance test of each device | True | The first performance test of each device is excluded, because for those tests we cannot define their previous measurement |
| Exclude consecutive test with *failed* outcomes | False | If this decision is made, then consecutive failures are excluded from the dataset, because the performance of the model may be perceived better than it truly is. For example, it turns out it is not hard to correctly predict that a performance test will fail if the previous test also failed. If the proportion of consecutive failures in the dataset is large, then this phenomenon can greatly impact the model performance, but the added value may be limited. Alternatively, the case could be made that actions will be defined per performance test anyway, so including consecutive failures is just a representation of the real-world scenario. |
| Filter tests where monitor type is unknown | False | Retrieving information that specifies the right type of monitor per device for a specific moment in time turned out to be difficult. Therefore, marking all entities unusable that have this missing data would greatly reduce the size of the final dataset. Therefore, this |

| | | |
|---|---|---|
| | | decision allows for the option to still include all data entities of which the monitor information is unknown and simply mark these fields as unknown. |
| Include Monitor replacement data | False | The extracted data related to monitor replacements looks incomplete. We cannot be sure of this, since there clearly would not be any data if there were no other replacements. However, this seemed unlikely. As a result, this decision is made to include none of this data, since it could be applied only to a part of the dataset. |

After all the data cleaning checks have been performed, the final dataset remains. Table 10 shows the fraction of data that has been excluded during the data cleaning process. It is important to be aware of this, because that gives an indication of the proportion of real cases this model can be applied to. Although the fraction of rejected data is quite high (approximately 50%), the dataset should still be sufficiently large to build a proper decision support model. As such, preference is given to a cleaner, but smaller dataset. Finally, we note that additional data collection efforts most likely lead to less data exclusion, but due to limited time the data collection efforts stopped here.

As discussed in Section 4.2.2, four data entities (Figure 11) are used to store data. Each of these entities can be marked usable or unusable, depending on the data cleaning decisions. The summarized outcome of the data cleaning process is shown in Table 10 and an example of a detailed data cleaning report generated by the Java application is shown in Appendix D.

*Table 10 - Overview of the fraction of data that is excluded after data cleaning.*

| Entity | usable | unusable | total | % usable |
|---|---|---|---|---|
| **Device** | 4527 | 2900 | 7427 | 61.0% |
| **Monitor** | 16457 | 12368 | 28825 | 57.1% |
| **Performance Test** | 55936 | 55268 | 111204 | 50.3% |
| **Subtest** | 950912 | 1003183 | 1954095 | 48.7% |

### 4.2.5. Data Output

Before any of the data can be used to build a decision support model, it must be formatted in the right way. In machine learning, a dataset generally consists of a certain number of rows and columns where each row represents a data instance (also referred to as a case or observation), and each column represents either a feature (input variable) or a class label. Since classification models predict the class label, it is important to define what each data instance and class label should represent. In this case, we have the option to represent the data instances as a subtest or as a complete performance test, since both options have an outcome that can be predicted to generate support for the decision to perform PM. Since the decision is made on the level of the performance test, we choose to represent each instance as a performance test in the basic model. However, Chapter 6 also discusses a model that uses subtests as data instances.

## 4.3. Data Overview

This section gives a brief overview of the final dataset that is used to develop the decision support model. Several characteristics of the data are highlighted that should be considered during the modeling phase. The dataset is the result of the decisions and considerations described in Section 4.2

In many binary classification applications, researchers and model developers must deal with the phenomenon of imbalanced datasets. A dataset is considered imbalanced when the difference between the proportions of class labels is high. A typical example of such a case is fraud detection where the goal is to detect fraud cases among many legitimate cases. Without any intervention, classifiers tend to predict most, if not all cases as legitimate cases, because that results in a very high model performance if you only consider the fraction of correctly predicted cases. However, this is generally not helpful. There are several ways to address this problem, but in any case, awareness of such an imbalance is of vital importance, because appropriate considerations can be made during the development of the model. Table 11 shows the class skew of the dataset for this project on four levels. Please note that in this overview a *device* or *monitor position* is considered to fail if they have had at least one performance test that has a failed class label. This is to illustrate that for a majority of the devices in the dataset, no performance tests have been performed that yielded a failed outcome. As becomes clear from the overview, the dataset is quite imbalanced. The basic model predicts on the level of the performance tests, which mean that approximately 92% of the cases in the dataset have a *passed* class label and only 8% have a *failed* class label.

*Table 11 - Class skew of the final dataset on four levels.*

| entity | passed | failed | total | % passed |
|---|---|---|---|---|
| **Devices** | 3,283 | 1,244 | 4,527 | 72.5% |
| **Monitor Position** | 14,084 | 2,373 | 16,457 | 85.6% |
| **Performance Test** | 51,655 | 4,281 | 55,936 | 92.3% |
| **Subtests** | 943,354 | 7,558 | 950,912 | 99.2% |

Another interesting observation is that performance tests tend to fail mostly due to a select group of subtests. As can be seen in Table 11, 85.3% of all subtests with a failed outcome are subtests that measure contrast. For the sake of clarity, all 17 subtests are grouped into four categories: *contrast*, *vignetting*, *contrast check* and *sharpness*. It is worthwhile to mention that the *contrast check* and the *sharpness* measurements rarely yield a *failed* result. Given that only 0.8% of the subtests fail (Table 11) and that the contrast checks and sharpness form only 1% and 2.7% of those failures respectively, it suggests that these subtests rarely ever contribute towards detecting systems that operate out of their predefined specifications. As mentioned before, a performance test yields a failed outcome if any of its constituent subtests yields a fail. However, there is of course the possibility that more than one subtest fails. Figure 13 gives insight into how often a performance test yields a failed outcome because more than one subtest has failed.

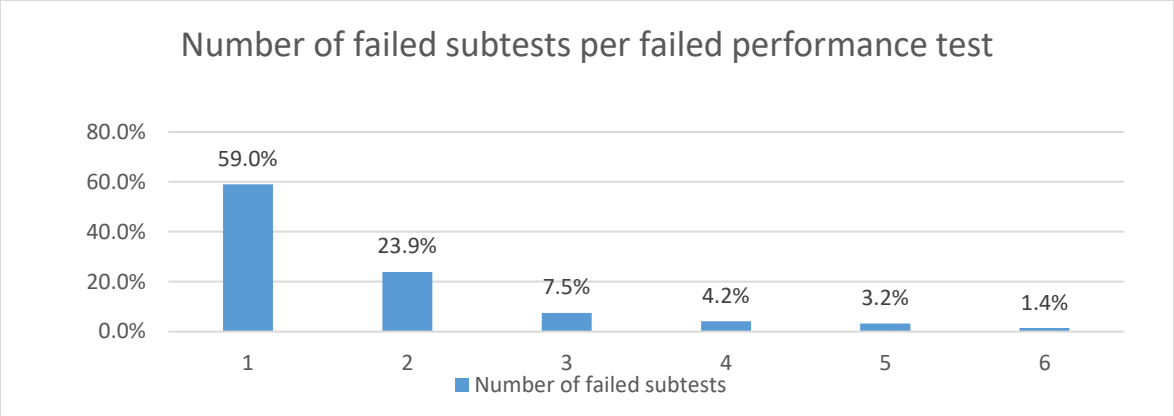



*Figure 12 - Number of failed subtests per failed performance test.*

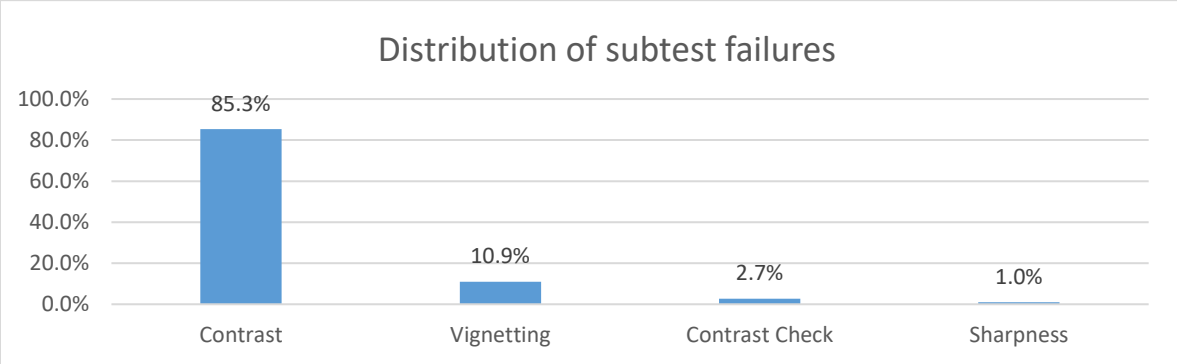



*Figure 13 - Distribution of subtest failures.*

Finally, it is worthwhile to consider how frequent failed performance tests are consecutive. As explained in Section 4.2.4, a significant proportion of all performance tests with a failed outcome are consecutive, suggesting that high performance may be misinterpreted, because while consecutive failures are easier to predict, it is perhaps not as valuable to do so. From a practical perspective, no distinction is made. Therefore, it is assumed justifiable to keep all failures in the dataset. However, excluding the consecutive failures from the dataset may give additional insight in the ability of the model to identify at what point performance tests are likely yield a failed outcome instead of a passed outcome. Table 12 gives a breakdown of how many failures are consecutive in nature, i.e. the number of performance tests with a failed outcome where the previous performance test on the same device and monitor also had a failed outcome.

*Table 12 - Information regarding the proportion of performance test with a failed outcome that are consecutive.*

| Consecutive | Non-consecutive | Total | % consecutive |
|---|---|---|---|
| 2,478 | 1,803 | 4,281 | 57.8 |

# 5. The Decision Support Model

This chapter is dedicated to the development of a decision support model. In this chapter the different aspects regarding the proposed decision support model are discussed. We note that we define the decision support model in generic terms and only in Chapter 6 is it applied to a specific case study. The chapter starts with an overview of the concept of a decision support model (Section 5.1). Section 5.2 is dedicated towards defining the binary classification problem in relation to the performance tests. Afterwards, a set of modeling assumptions are discussed in Section 5.3 that are used to construct the decision support model. In Section 5.4, attention is given towards selecting the right performance metrics. Next, the chapter briefly touches on the concept of k-fold cross validation as a method to assess the randomness of the model in Section 5.5. Then, Section 5.6 discusses the random forest algorithm and why it is useful for this project. Finally, the chapter ends with discussing how the prediction of the classification model can be transformed into a suggested course of action given the operational context to which the model is to be applied.

## 5.1. Concept Decision Support model

In this section we outline the structure of the proposed decision support model. The model is separated into two parts. The first part consists of a binary classification algorithm. Its purpose is to predict the system's condition on the aspect that the performance test evaluates. It takes a feature vector $x$ and generates a prediction in the form of a *score* $s(x)$, representing the probability estimate of the model that the system is faulty. The second part of the decision support model is basically a decision rule considering the probability estimate of the classification model as well as the operational context, defined by operating conditions and constraints, to generate a suggested course of action. The concept of the decision support model is shown in Figure 14.

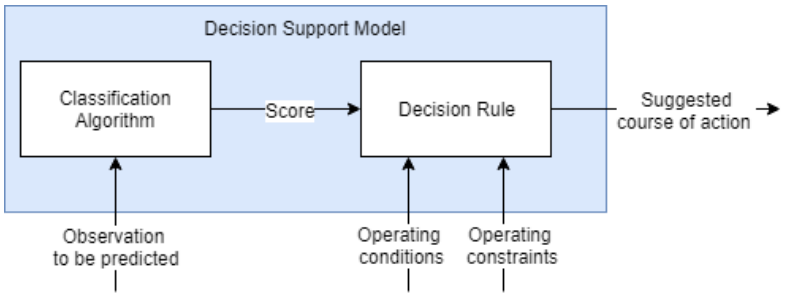



*Figure 14 - Abstract overview of the decision support model.*

One of the operating conditions is based on costs associated with misclassifying the different class labels, referred to as the cost skew (see Section 5.7). As can be deduced from Figure 14, the operating conditions are specifically used in a separate decision rule and not in the classification algorithm. Therefore, the binary classification algorithm is not cost-sensitive, because the learning process is not influenced by the cost parameters. However, the decision support model as a whole is considered cost-sensitive, because cost factors do play a role in the outcome of the model trough the decision rule. It could be argued that the use of a cost-sensitive learning algorithms, discussed by He & Garcia, (2009) and Domingos, (1999) effectively pursues the same goal, i.e. influence the decision making (or classification) based on cost incentives. These cost-sensitive classification algorithms have not been chosen for several reasons. First, in cost-sensitive learning methods it is implicitly assumed costs are known beforehand, because they are required to train the classification model. This was not the case during this research. Furthermore, if the

cost skew changes, the model must be retrained, because the costs influence the learning process of the model. Given that costs may be different for different application areas (for example countries), this could lead to a range of models instead of a single model. In conclusion, the key advantage of the proposed decision support model is that the classification algorithm can be constructed without knowing any of the operational conditions and constraints upfront and that it is easily applicable to situations where operating conditions, including costs, tend to change.

## 5.2. General Formulation of the Classification Model

Classification can be described as a task of identifying to which of a set of categories a data instance belongs, given a set of training instances of which the category is known. Each data instance is defined as a vector $x = \{x_1, x_2, \dots, x_n\}$ of $n$ features and a class label $y$. Then, an implementation of a classification algorithm, referred to as a classifier, aims to predict the class label $\hat{y}$ given vector $x$ by considering training instances where both $x$ and $y$ are known. As explained in Section 4.2.5, in our research, a data instance refers to a performance test that is conducted on a medical IXR device. Here, the feature vector $x$ represents characteristics regarding the medical device and the performance test and the class label $y$ represents the outcome of the performance test. Note that this classification problem is binary in nature, because there are only two possible class labels: *passed* and *failed*. For binary classification, it is common practice to define the class labels as positives and negatives, because that allows for the use of the standard terminology. Therefore, the set of class labels for this classification problem is defined as $Y = \{+, -\}$ where $+$ represents a performance test with a *failed* outcome and $-$ represents a performance test with a *passed* outcome.

For a given classifier, each data instance in the validation set (sometimes referred to as the test set) has a real class label $y$ and can be assigned a predicted class label $\hat{y}$. To assess how well a classifier performs, the differences between $y$ and $\hat{y}$ are examined. For this purpose, four basic counts are defined:

- $TP$: Number of correctly predicted real positives $(y = +, \hat{y} = +)$.
- $TN$: Number of correctly predicted real negatives $(y = -, \hat{y} = -)$.
- $FP$: Number of wrongly predicted real negatives, i.e. false alarm or type 1 error $(y = -, \hat{y} = +)$.
- $FN$: Number of wrongly predicted real positives, i.e. a missed hit or type 2 error $(y = +, \hat{y} = -)$.

Typically, these four counts are expressed in a confusion matrix for a quick overview of the classifier's performance. Table 13 shows the confusion matrix for our classification problem with each combination of $y$ and $\hat{y}$. In addition, Table 13 shows how the class labels for $y$ and $\hat{y}$ should be interpreted in relation to the practical classification problem for the performance test. In section 5.4 these four counts are used to select appropriate quantifiable performance measures to evaluate different classifiers.

*Table 13 - Confusion matrix for performance test classification.*

| | $\boldsymbol{y = +}$ **real positive (system faulty)** | $\boldsymbol{y = -}$ **real negative (system ok)** |
|---|---|---|
| $\boldsymbol{\hat{y} = +}$ **classified positive (perform test)** | $TP$ | $FP$ |
| $\boldsymbol{\hat{y} = -}$ **classified negative (skip test)** | $FN$ | $TN$ |
| | $P = TP + FN$ | $N = FP + TN$ |

## 5.3. Modeling Assumptions

During the development of the model, several assumptions are made. These assumptions are listed and discussed below.

1. The outcome of the performance test is used to determine if the system was operating outside the allowed, predefined specifications at the time the performance test was conducted.
2. The outcome of the performance test is defined by the outcome of its constituent subtests. If one or more of the subtests yields a *failed* outcome, then the performance test yields a *failed* outcome too. Otherwise, the performance test yields a *passed* outcome.
3. If the performance test yields a *passed* outcome, then it is assumed that the system was operating correctly at that point in time, regardless of any potential actions performed by the FSE to reach that result.
4. The final dataset discussed in Chapter 4 is representative of the original data.

The first assumption implies that the decision to label a system as (not) faulty (with respect to the aspect that the performance test checks) is purely dependent on the outcome of that performance test, regardless of any other circumstances that may have influenced the outcome of the performance test. This is a fair assumption, since in the current situation the performance test is also decisive for any follow-up maintenance actions, regardless of the true ability of the performance test to measure if the system is actually faulty. In addition, data related to performance tests that show strong indications of inaccurateness are already excluded from the dataset.

The second assumption states that the outcome of the performance test is purely dependent on the outcome of its constituent subtests. This is because the performance test itself is not a single measurement, but a combination of quality assurance measurements called subtests. As such, the outcome of the performance test as a whole must be determined based on the constituent subtests. The relation described in the assumption is equivalent to the real-life situation, with the exception that the outcome of the performance test in the database of Philips is determined based on the subtests from all relevant components in the system. In the case of the monitor performance tests, this means that subtests are conducted for each monitor and if any subtest of any monitor yields a *failed* outcome, the whole

performance test for that device yields a *failed* outcome. Since the decision support model in this research is designed to make predictions per relevant part, i.e. per monitor, the definition of the outcome of a performance test is changed such that it only considers quality assurance measurements related to the relevant part. A prediction per monitor is more beneficial, because when a monitor is considered faulty, only that monitor must be tested and not the all the monitors connected to the medical equipment.

The third assumption is critical, but necessary. It implies that the outcome of the performance test is independent from any actions a field service engineer may have performed during that planned maintenance visit. If this assumption is not made, then the situation could exist where a performance tests yields a *passed* result, because the engineer performed certain actions. Then the implicit assumption that a performance test with a *passed* outcome is equivalent to an unnecessary performance test will not hold anymore. It is not known how often this happens, but it is unlikely that the occurrence is so frequent it notably undermines the analysis. Either way, this assumption is critical, because although it may hold for the monitor performance test discussed in Chapter 6, it may not hold for other performance tests.

The final assumption implies that after all the original data has been cleaned, the remaining dataset is still representative of the original data. This is important, because only the cleaned data is used to construct the model, but ideally, the model is applied to all operational systems, including those that may have been excluded, because no appropriate data was available to construct the model. This is a fair assumption, given that no apparent reasons have been found that indicate otherwise. Chapter 4.2.4 discusses the data cleaning process including justification for several data cleaning decisions.

## 5.4. Model Performance Metrics

In order to justify that the right classifier is used for the decision support model, we must choose an appropriate way to evaluate the performance of the classification model. As mentioned in Section 5.2, the performance of each classifier can be expressed by using a confusion matrix. However, to give a more meaningful interpretation of the performance of a classifier, several other performance metrics are commonly used (Powers, 2011) . Table 14 shows a selection of the most widely used measures. All these performance metrics can be expressed in terms of the four counts that are presented in the confusion matrix.

*Table 14 - Widely used performance metrics for binary classification.*

| **Performance measure** | **Alternative labels** | **formula** | **Description** |
|---|---|---|---|
| Accuracy | | $\frac{TP+TN}{TP+TN+FP+FN}$ | Proportion of correctly classified instances |
| True positive rate (TPR) | Recall, Sensitivity | $\frac{TP}{TP+FN}$ | Proportion of positives correctly classified as positives |
| Precision | Positive predictive value | $\frac{TP}{TP+FP}$ | Proportion of true positives from the set of classified positives |
| True negative rate (TNR) | Specificity | $\frac{TN}{TN+FP}$ | Proportion of negatives correctly classified as negatives |
| False Positive Rate (FPR) | Fall-out | $1-TNR$ | One mines the true negative rate |

Depending on the context to which the classification model is applied, different performance measures may be appropriate. Section 5.4.1 and Section 5.4.2 elaborate on the performance metrics in Table 14 and discuss why they are appropriate or inappropriate for this research.

### 5.4.1. Accuracy

Arguably, accuracy is the most intuitive and basic performance metric, because it simply states the proportion of how often the model predicted a case correctly. For this research problem however, the usefulness of this performance metric is limited, because it does not consider the relative importance between correctly classifying positives and negatives. In situations where datasets are highly imbalanced, the model could reach a high accuracy by simply classifying all cases equal to the dominant class in the dataset. To illustrate this, consider our case where 92% of the performance tests are negative and only 8% is positive. A classification model that classifies everything as a negative could reach an accuracy of 92%. Without considering any contextual details, one could say 92% accuracy is a respectable performance. However, the classification model shows no intelligence whatsoever, since it classifies everything the same. Now, this would not matter if equal importance is given to correctly classifying each of the classes. However, in many scenarios, including this problem, the cost for misclassifying a different class labels may be considerably different. Therefore, one can clearly see that accuracy is not adequate to evaluate the model performance for our classification problem.

### 5.4.2. Combination of metrics

Other commonly used performance metrics are precision, recall and Specificity. Precision and recall are often used together to evaluate the performance of classification tasks in the context of information retrieval. In such tasks, the precision metric is particularly interesting, because it gives insight into what fraction of returned samples is actually relevant. However, both precision and recall do not consider the performance of identifying negative labels. For that reason, the combination of precision and recall is limited for our problem too. In our problem, costs are associated to classifying any of the class labels wrong. Therefore, a combination of performance metrics is required that evaluates the performance for positive labels as well as the performance for negatives labels.

Another commonly used set of performance metrics to evaluate the performance of binary classifiers is the *True Positive Rate* ($TPR$) and the *True Negative Rate* ($TNR$), sometimes referred to as *sensitivity* and *specificity* respectively. Both metrics are the same except for the class label they apply to. $TPR$ expresses the fraction of correctly classified positives from the real set of positives and $TNR$ expresses the fraction of correctly classified negatives from the real set of negatives. The combination of $TPR$ and $TNR$ works well for our research problem for several reasons. First, almost all changes in the confusion matrix influence at least one of the metrics. Sokolova & Lapalme (2009) performed an analysis on the ability of a performance metric to preserve its value under a change in the confusion matrix. In total, they define eight invariance properties, i.e. ways a confusion matrix may change. Their research shows that for each invariance property at least one of the metrics is non-invariant, meaning that a change in the confusion matrix analogous to their defined invariance properties is reflected in at least one of the two metrics. Secondly, it can be argued that both $TPR$ and $TNR$ are not directly related to the class skew, because both metrics are defined over a real set (Korst, Pronk, Barbieri, & Consoli, 2018) . Furthermore, they argue that consequently the class skew of a training set need not be the same as the class skew during operation. This is fortunate, because for some classification algorithms it is said that the use of a balanced training set will give better results. In fact, Chapter 6 reveals how we make use of this by influencing the

imbalanced dataset during the training of the classification model. Finally, as will become clear in the remainder of this chapter, the use of $TPR$ and $TNR$ allow us to tweak the model in such a way that it is optimized with respect to several operating conditions, discussed in Section 5.7.

Although using $TPR$ and $FPR$ to express the performance of the model is useful, the downside is that it consists of two metrics. Therefore, when comparing performances of different model instances, it may not be clear what performance is superior. As is explained in Section 5.7, many different combinations of TPR and FPR can be obtained from a single classifier and these can be expressed in ROC (*receiver operating characteristic*) space. A metric that summarizes the combinations of $TPR$ and $FPR$ is the *Area Under the Curve* ($AUC$) performance metric. It is simply the area under the classifiers curve as a fraction of the unit square. Provost & Fawcett, (2013) argue that it can be useful to use this metric when the operating conditions are not known. In other words, when there is no information available as to what combinations of $TPR$ and $FPR$ are optimal, a single metric that captures the performance over the whole space may be more useful. Since the $AUC$ captures the performance along the whole ROC curve, differences at specific points in ROC space may only have a small impact on $AUC$. Consequently, small differences (or even similar values) for $AUC$ do not necessarily translate to the same model performance.

In the remainder of this research, the primary method to evaluate the performance of model is a plot of the ROC curve in ROC space. Since the ROC curve represents all the trade-offs that can be achieved for TPR and FPR, it essentially captures the performance of the binary classifier independently from the decision rule. To quantify the ROC curve, the AUC metric is used. Although visual interpretations of ROC curves generally provide more insight than the quantified AUC metric, it is still useful to also quantitatively compare different model outcomes. Finally, if single points on the ROC curve are to be evaluated (as a result of determining the operational context that lead to the formulation of the decision rule), TPR and FPR are used.

## 5.5. K-fold Cross validation

Section 5.4 introduced several performance metrics that we use to assess the performance of the model. However, it would be naïve to assess the performance of a model with only one single model instance. Instead, it is good practice to train and test the model several times to gain insight in how consistent the model performs. To gain insight into the consistency of a model, resampling methods are an essential tool, because they can be used to construct different model instances based on the same dataset.

Resampling methods work by repeatedly drawing samples and fitting a model to each sample. Then the difference between all the fits is examined to extract information that would not be available when a model is only fitted once. Particularly the variation between the fitted models is of interest as that gives insight into the randomness of the model.

A popular resampling method is *K-fold cross validation* (James, Witten, Hastie, & Tibshirani, 2013). In this resampling method, the data is randomly split in $k$ groups, or folds, of approximately equal size. Then each group $k$ is used as a validation or test set and the remaining $k - 1$ groups are used to train the model. Cross validation is used to test how well the model behaves using a given training set to predict data it has not yet seen (test data).

It is important to note that the decision for $k$ is only relevant for the evaluation of the model performance, but not for the final model that may be used in practice. This is because the dataset is split randomly into $k$ groups, where each group is used once for testing a trained model. Consequently, using k-fold cross

validation leads to $k$ models, using only the fraction $\frac{k-1}{k}$ of the dataset. Then these $k$ models may be compared to see if performance is consistent. However, the model that may be used in practice can use 100% of the training data, because at that point the model is already validated. Hence, the model performance in practice is independent of $k$, albeit that the value $k$ may have influenced the process of the modeler to set appropriate hyperparameters and make appropriate modeling decisions.

In the remainder of this research k-fold cross validation is used to evaluate if models perform in a consistent manner. James, Witten, Hastie, & Tibshirani, (2013) argue that using $k = 5$ or $k = 10$ is generally a good decision. Therefore, we use $k = 10$.

## 5.6. Choosing a Classification Algorithm: Random Forests

In Section 5.1 the concept of the proposed decision support model is discussed. We argued that an important part of the decision support model consists of a binary classification algorithm. In this section we discuss one possibility for the binary classification algorithm: *random forests*.

### 5.6.1. Importance of the Classification Algorithm Decision

Note that even though only one classification algorithm is discussed here, it could be replaced relatively easily by other classification algorithms. As long as a classification algorithm can output a binary probability estimate, it should fit within the proposed decision support model. The reason only one type of binary classification algorithm is considered, is primarily because the purpose of this research is not to come up with the single best model for a specific classification problem. Rather, the purpose is to propose a decision support model that is applicable to the stated research problem, i.e. applicable to multiple performance tests. It is entirely possible that other binary classification algorithms result in better performance, but in a sense that is also of secondary importance, because it may very well vary between different performance tests. Therefore, the decision to use a particular binary classification algorithm is considered to be part of the implementation of the proposed decision support model instead of a fixed decision in the proposed decision support model. As such, Section 5.6.3 discusses why the random forest algorithm is chosen for the presented case study in Chapter 6.

### 5.6.2. Random Forest algorithm

The first random forest algorithm was developed by Tin Kam Ho (Ho, 1998) and was later extended by Leo Breiman (Breiman, 2001). The random forests algorithm is an *ensemble learning* method and can be used for both classification and regression. Ensemble methods refer to the use of multiple classifiers to obtain better predictive performance than its constituent individual learning algorithms typically achieve. For random forests, the individual learning algorithms are *decision trees*. In essence, the random forest algorithm works by constructing a multitude of decision trees and then uses majority voting to perform classification. As the name suggests, the forest is constructed randomly in the sense that a random sample of observations along with a random vector of features are used to build each tree. This is to stimulate the construction of different, less correlated trees. The output of the random forest can be expressed as a matrix of scores $s^{\hat{y} \in Y}(x)$, where each row represents a case in the test set and each column represents the score for a certain class label, i.e. the fraction of decision trees that voted for that class label. Since binary classification knows only two class labels, it is easier to just refer to $s(x)$ to denote $s^{+}(x)$, because $s^{-}(x)$ directly follows from $1 - s^{+}(x)$. In classification problems with more than two class labels this would not work. Based on the score $s(x)$, various decision rules can be defined that determine how the scores are transformed into classification. The simplest and widely used decision rule for random forests

is majority voting, i.e. the class label with the highest score is assigned. However, Section 5.7 presents an alternative way to use $s(x)$, which is also used for our classification problem. For more information on the inner workings of random forests the reader is referred to the original work by Leo Breiman (Breiman, 2001).

Section 5.5 briefly described the need for resampling methods to gain insight into the variation between model instances and their performance. However, it turns out that there is a straightforward way to estimate the test error of a random forest model without the need for cross-validation. This is called *out-of-bag* (OOB) error estimation. It is easy to show that when a decision tree is constructed in a random forest, roughly 2/3 of the samples are used to construct the tree, leaving 1/3 of the observations unused, which could potentially be used to test the decision tree. This is what OOB Error Estimation uses. Given that we use stratified sampling, we specifically assign the number of samples drawn to construct each tree. In our case, this results in a considerably different ratio between the training and test set (roughly 1/100 training set and 99/100 test set). Although OOB error estimation is useful for setting hyperparameters, we primarily rely on k-fold cross validation, because it gives us more options when comparing different model instances.

### 5.6.3. Advantages of Random Forests

In Section 5.6.1 we argued that the several different classification algorithms could be used in the decision support model, but that we discuss one particular classification algorithm to demonstrate the model with a case study (Chapter 6). In this section we briefly motivate why we consider random forests to be a good decision for this particular research problem.

There are several reasons to use random forests as the classification algorithm for this research problem. First, as Leo Breiman points out, the random forests are quite good in handling noise and outliers in the data. Even though many outliers have been excluded in the data preparation phase, it is likely outliers remain. Secondly, random forests are rather transparent in contrast to certain other machine learning algorithms, such as neural networks. This helps in understanding how the model behaves and consequently helps in understanding how to tweak the model by setting appropriate values for the hyperparameters. Third, if random forests are used for binary classification, there is a convenient way to influence the bias towards classifying positives or negatives, by simply introducing a threshold $T$. This is important, because that allows us to tweak the model according to a set of operating conditions, explained in Section 5.7. An additional advantage of random forests, although not as relevant in this case, is that the algorithm computations are easily parallelized, since all trees are created independently. Therefore, large computations can be scaled easily for bigger models or datasets. Finally, there was good prior understanding of the random forests learning methods within the department of Philips, which helped accelerate the development of the whole decision support model.

Aside from the aforementioned reasons, several studies show that random forests tend to perform well. A study on the performance of learning algorithms (Caruana, Karampatziakis, & Yessenalina, 2008) found that random forests showed the highest overall performance across all included problems and metrics. The study compared ten popular learning algorithms on 3 metrics and 11 datasets. In addition, another study that studied the $AUC$ performance of classification algorithms for imbalanced datasets (Brown & Mues, 2012) concluded that gradient boosting and random forest led to the best results. They found that this was true especially when datasets were more imbalanced. Although we are keenly aware that two studies are not irrefutable proof that random forests are a superior learning algorithm, the combination

of the aforementioned arguments and these studies does provide adequate motivation to use them for this the case study presented in Chapter 6.

## 5.7. ROC Space, Threshold and Operational Context

As explained in Section 5.1 the concept of the proposed decision support model consists of two parts: the classification algorithm and the decision rule. Section 5.2 already explained the role of the classification algorithm. This section is dedicated to how the decision rule is determined as well as how its relation to the classification algorithm. In particular, three topics are covered: the ROC space, the definition of a Threshold $T$ and the role of operating conditions and constraints.

As explained in Section 5.2, the performance of every classifier can be expressed by a confusion matrix from which several performance metrics can be derived. $TPR$ and $FPR$ are often visually presented in the receiver operating characteristic (ROC) space. The y-axis represents the $TPR$ and the x-axis represents the $FPR$, which is equal to $1 - TNR$. From now on, the $FPR$ is used instead of $TNR$, because this is the default metric in ROC space. Figure 15 shows an example of a ROC space with the performance of several classifiers expressed by their $TPR$ and $FPR$.

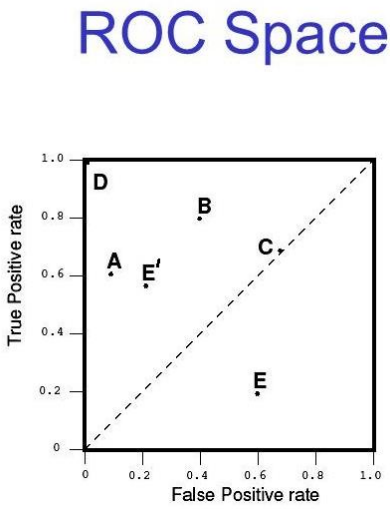



*Figure 15 - Five classifiers (A, B, C, D, E) expressed as a point in ROC Space with (FPR, TPR).*

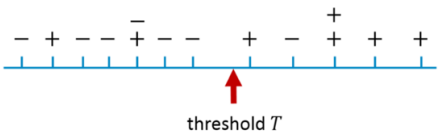
In Figure 15, five arbitrary classifiers are presented. Classifiers $A$ and $B$ are relatively normal classifiers where classifier $A$ has a slightly worse $TPR$ than classifier $B$, but it has a considerably lower $FPR$. For classifier $C$, $TPR = FPR$ ($C$ lies on the dotted line in Figure 15), which is a special situation. In this case, similar performance can be obtained without the use of a classification algorithm. For example, if $TPR = FPR = 0.5$ is desired, randomly classifying half of the observations as positives should result in a correct classification of half the real positives. Analogous to this example, every point on the dotted line in Figure 15 can be achieved without the use of a classification algorithm by simply classifying a fraction of samples equal to the desired $TPR$ as positives. Classifier $D$ is represents a perfect classifier that has classified every observation correctly, resulting $TPR = 1$ and $FPR = 0$. In practice, this is rarely achievable. Finally, Classifier $E$ lies below the dotted line. Therefore, following the reasoning given for classifier $C$, this algorithm performs worse than randomly classifying observations. However, if the outcome of classifier $E$ is turned around, i.e. predicted positives are changed into negatives and vice versa, then $E$ would be positioned in ROC space as if it was mirrored around the dotted line, see $E'$. In short, the outcome of

classifiers that consistently perform worse than random classification can be negated to achieve the opposite performance.

Recall that the performance of a random forest binary classifier for a given validation set can be expressed by a matrix with two columns: $s(x)$ and $y$. Here, $s(x)$ is the fraction of trees that would classify the observation as positive based on the feature vector $x$ and $y$ is the real class label of the observation. By default, random forests use majority voting to assign the class label $\hat{y}$. In essence, this is done by defining a threshold value $T$ with $T = 0.5$. Then each class label $\hat{y}$ is assigned as follows:

$$\hat{y} = \begin{cases} + \ if \ s(x) \geq T \\ - \ if \ \ s(x) < T \end{cases}$$

From this equation, it follows that $T$ affects how observations are classified. Consequently, for different values of $T$, different confusion matrices and different combinations of $TPR$ and $FPR$ are obtained. Note that if $T$ is increased, more data instances will be assigned a negative class label $(\hat{y} = -)$, decreasing both $TPR$ and $FPR$. Alternatively, if $T$ is decreased, more data instances will be assigned a positive class label $(\hat{y} = +)$, increasing $TPR$ and $FPR$. Figure 16 illustrates how this trade-off works.




| $s(x)$ | $y$ | $\hat{y}$ |
|---|---|---|
| 0.02 | − | − |
| 0.06 | + | − |
| 0.12 | − | − |
| 0.21 | − | − |
| 0.37 | + | − |
| 0.37 | − | − |
| 0.43 | − | − |
| 0.56 | − | − |
| 0.67 | + | + |
| 0.72 | − | + |
| 0.84 | + | + |
| 0.89 | + | + |
| 0.89 | + | + |
| 0.91 | + | + |
| 0.95 | + | + |

Threshold $T$

*Figure 16 - Example of how different values of $T$ lead to different a classification of $\hat{y}$.*

In Figure 16, values for $s(x)$ are ordered in ascending order. Then it becomes clear that as $T$ decreases, more observations will be assigned $\hat{y} = +$ and thus $TP$ increases or stays constant. Since $TPR$ is defined as $TPR = TP/P$, $TPR$ can never decrease as $T$ decreases. However, the same is also true for $FPR$, resulting in a trade-off between the two metrics.

We note that $T$ can not be changed such that both metrics improve, i.e. higher $TPR$ and lower $FPR$. Therefore, $T$ is an input parameter that can influence the trade-off between $TPR$ and $FPR$. In addition, only a finite number of values for $T$ are relevant, because it only makes sense to define new values for $T$ that lead to either an increase $TPR$ or a decrease in $FPR$. Therefore, this trade-off can also be represented with a set points $(FPR, TPR)$ in ROC space. Finally, the convex hull is taken over all the points in the ROC

space, such that only relevant points remain. Then all the points in the ROC can be presented as a convex ROC curve. Figure 17 shows an example of how such a ROC curve might look.

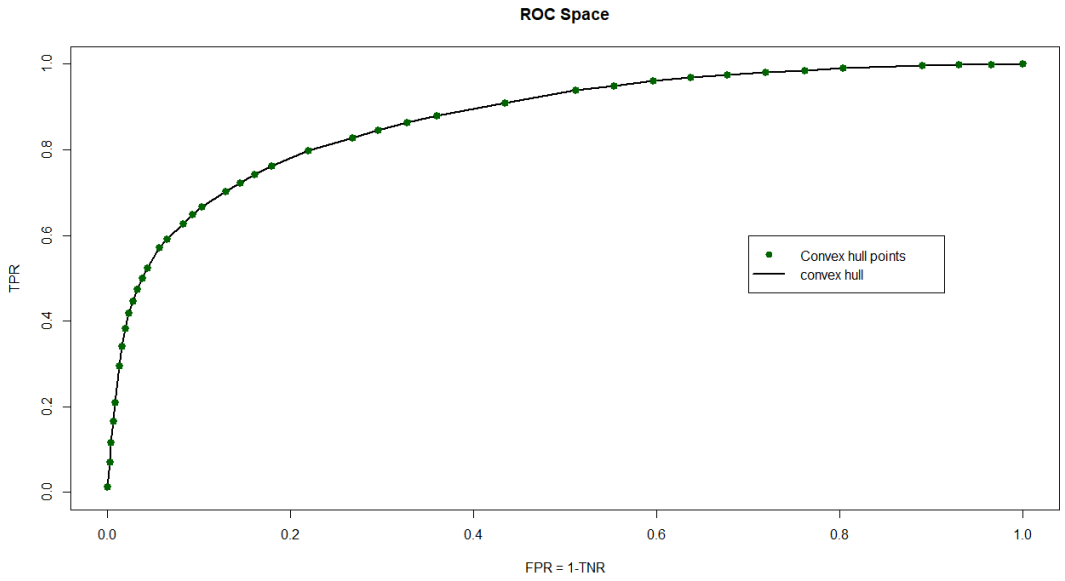



*Figure 17 - Example of an ROC curve.*

It is worthwhile to mention that the trade-off between $TPR$ and $FPR$ is likely to be more extreme for low or high values of $TPR$ and $FPR$. From Figure 18 it can be observed that for high and low values of $TPR$ the slope is relatively flat or steep respectively. The slope at a point on the ROC curve essentially defines the trade-off. Therefore, we note that for a certain $T$ the trade-off can be quantified in the following way.

$$Tradeoff(T) = \frac{\Delta TPR}{\Delta FPR}$$

Now the question is what point on the ROC curve is optimal. In other words, what value of $T$ leads to the best trade-off? To answer this question, the operational context of where the model is to be applied must be considered. In the work of Korst et al. (2018), two operating conditions, based on the work of Provost & Fawcett (2001) and Fawcett (2006), are discussed that are of interest to this problem. They are defined as follows:

- **Class skew**: defines the difference between the fraction $p(+)$ of positives and the fraction $p(-)$ of negatives.
- **Cost skew**: defines the difference between the cost $c(+|-)$ of a false alarm and the cost $c(-|+)$ of a missed hit.

The authors continue by showing that these operating conditions can be combined into a single parameter: *the cost slope*. This is given by the following equation.

$$cost\ slope = \frac{p(-) \cdot c(+|-)}{p(+) \cdot c(-|+)}$$

Finally, they show that to choose the point with the smallest expected cost on a given ROC curve, the following observation holds

$$slope_{right} \leq \frac{p(-) \cdot c(+|-)}{p(+) \cdot c(-|+)} \leq slope_{left}$$

Where $slope_{left}$ is the slope of the line segment connecting to the left of the classifiers ROC point and $slope_{right}$ is the slope of the line segment connecting to the right of the classifiers ROC point.

At this point, all the required elements are defined to determine which point of the ROC curve is best with respect to the operating conditions. However, the operating conditions only characterize the operational context in terms of cost. Many of the PM activities for IXR devices are also tied to safety concerns. For that reason, we also introduce two operating constrains besides the operating conditions.

- **Minimum TPR ($\boldsymbol{TPR_{min}}$)**: defines the minimum $TPR$ the classification model must adhere to. This operating constraint is introduced to allow certain safety constraints to dictate what fraction of false negatives, i.e. faulty systems that are not identified as faulty, is allowed. This constraint is relatively easy to implement in the model, because all points on the ROC curve that do not meet this constraint are simply considered unfeasible and consequently excluded. However, it should be noted that this may lead to a situation where the optimal point on the curve with respect to the operating conditions may not be feasible. In that case, the closest feasible point on the ROC curve is taken.
- **Maximum untested timeframe ($\boldsymbol{\tau_{max}}$)**: defines the maximum amount of time allowed between the execution of two performance tests for a given system. In other words, if a decision to skip a performance test leads to a situation that would result is the system being untested for a period greater than $\tau_{max}$, the performance test is performed anyway. As soon as this constraint is violated, the classification score $s(x)$ becomes irrelevant. Therefore, this constraint cannot be captured in ROC space, because overrides the decision of the decision support model.

Together, these operating constraints add value to the model in the sense that they account for certain safety related aspects, which cannot be defined in terms of costs. For example, regulation may state that a system must be tested every two years. Even without regulation it is probably not desirable to let the model decide to skip a performance test many times in a row. In a similar way, safety experts may argue that such a model is only acceptable if the $TPR$ reaches a certain level. Depending on the severity of these safety concerns, appropriate operating constraints can be set, such that unwanted scenarios can be avoided. Figure 18 shows an example of how the operating conditions and operating constraints can be visualized in ROC space.

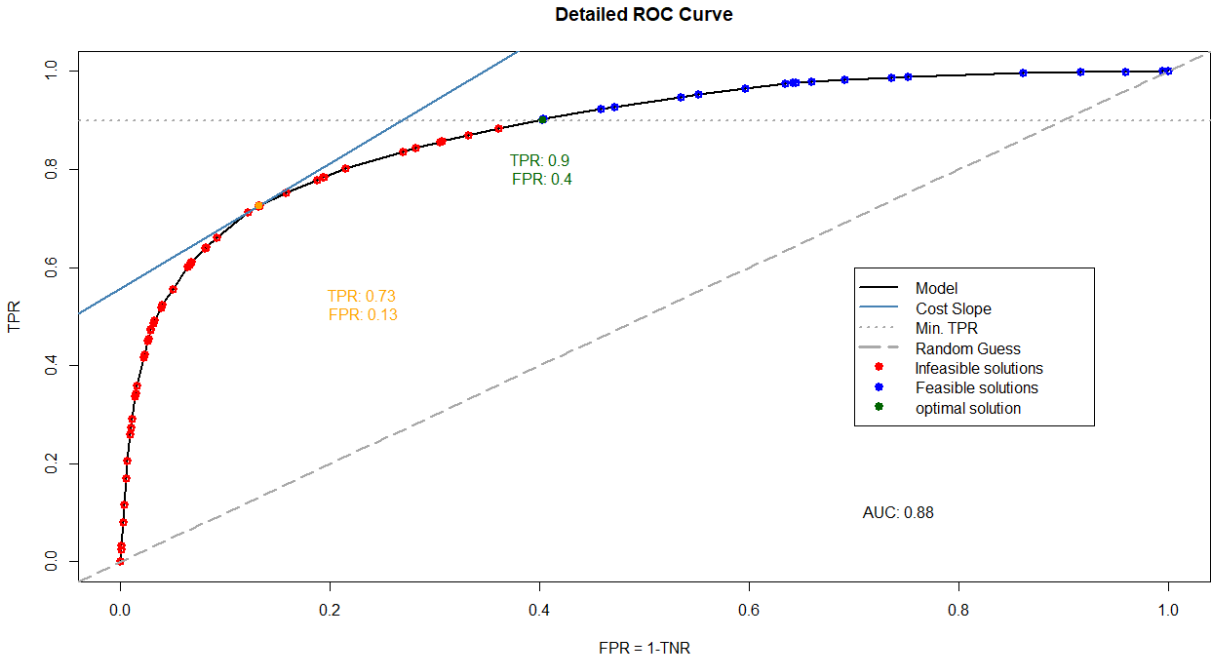



*Figure 18 - Example of a ROC curve with a cost sloped based on (arbitrary) operating conditions and a minimum TPR.*

# 6. Case Study: The Monitor Performance Test

This chapter discusses the implementation of the generic model described in Chapter 5 for a specific performance test: *the monitor performance test*. The model is implemented in R, using the *randomForest* package (Liaw & Wiener, 2002). First, the basic model is discussed, and motivation is given for the chosen default hyperparameters and modeling decisions in Section 6.1. Then, subsequent sections 6.2 through 6.4 discuss how different decisions impact the model performance. Section 6.2 is dedicated to selecting the right model hyperparameters and how important they are for model performance. Next, we consider the importance of several features and study if better performance can be gained by using a subset of the available features in Section 6.3. In Section 6.4, the consideration of two datasets with a different decision regarding data selection is discussed. Subsequently, Section 6.5 introduces a model extension that implements a two-stage learning algorithm, instead of a one-stage learning algorithm in an attempt to achieve better performance. The chapter ends with a validation on newly obtained data since the construction of the model. This Functions as a final validation of the model to confirm it does not (severely) suffer from overfitting.

## 6.1. Basic model

The model described in Chapter 5 can be implemented in variety of ways, because in addition to the model hyperparameters, there are several other modeling aspects that require a decision. We define the basic model as the implementation of the generic model described in Chapter 5, with certain chosen default decisions for the modeling aspects. These modeling aspects are briefly described in Table 15. In Section 6.2 through 6.4 these modelling aspects are discussed and their role on the performance of the model is evaluated. To extract meaningful conclusions, the basic model is used as a benchmark to compare all models with several different decisions for each modeling aspect.

*Table 15 - Modeling aspects that require a decision.*

| **Modeling aspect** | **Explanation** |
|---|---|
| Model hyperparameters | A random forest algorithm has several model hyperparameters that can be changed to obtain different models. This excludes the operating conditions and constraints discussed in Chapter 5, because they do not influence how the classification model is constructed. |
| Feature selection | In total, 63 features are available that can be used to construct the random forest. However, not all features contribute towards more predictive power. Therefore, the selection of several features is important to consider. |
| Dataset selection | Several decisions are made in the data preparation stage that can influence the model performance and the interpretation of the model performance. Since options are numerous, we only consider the most important decision, which is to include or exclude performance tests with consecutive failures. |

Table 16 shows the default decisions for the modeling aspects for the basic model. Each is briefly motivated.

*Table 16 - Model decisions and hyperparameters chosen for the basic model.*

| **Modeling aspect** | **Decision** |
|---|---|
| Model hyperparameters | • $nTree = 1000$<br>• $mTry = \sqrt{(number\ of\ used\ features)} = \sqrt{63} \approx 8$<br>• $Stratified\ sampling\ size\ (+,-)\ = (450{,}450)$ |
| Feature selection | All features are used in the basic model. |
| Dataset selection | Consecutive failures are included in the dataset. In addition, the default decisions regarding the data preparation are used as described in Section 4.2.4 |

Table 16**Error! Reference source not found.** shows the performance of the classification model in ROC space. The cost slope is included as well to illustrate the full purpose of the model, but note that this slope is based on operating conditions, which only becomes relevant after the construction of the classification model. Therefore, a realistic, but arbitrary cost slope is used. It is also worthwhile to mention that when a reference is made to the performance of a model, i.e. a particular method for describing how some input data relate to what we aim to predict, it essentially means the performance deduced from the aggregate of $k$ model instances due to k-fold cross validation.

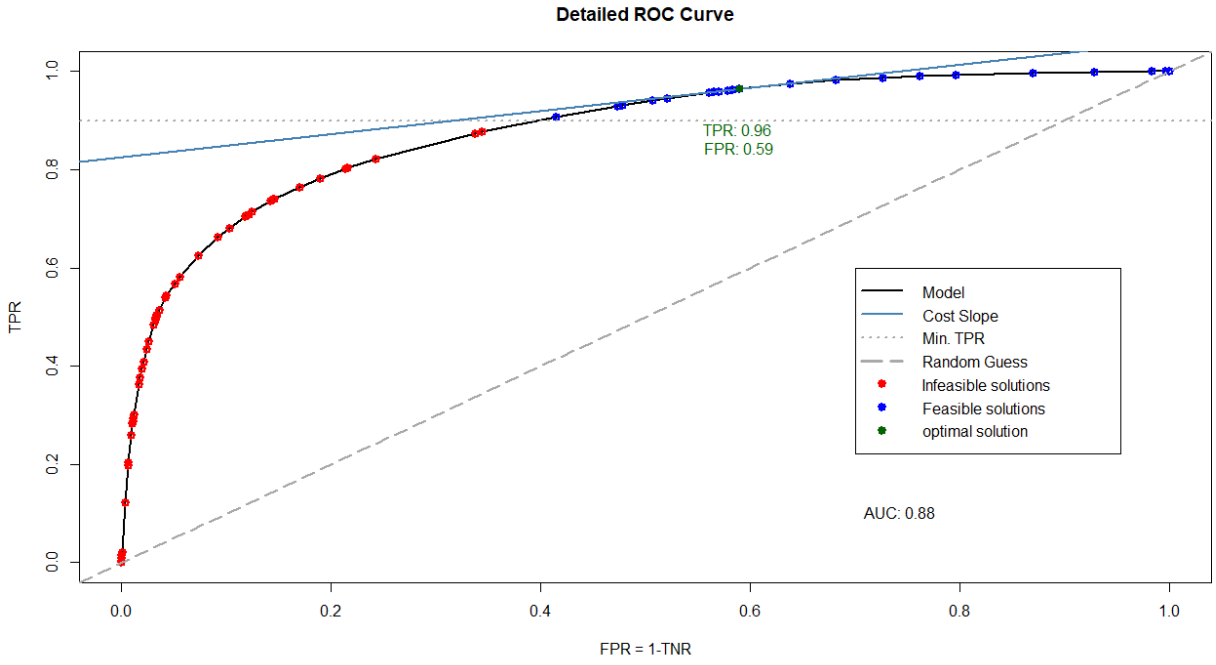



*Figure 19 - Performance of the basic model expressed in ROC space.*

In Chapter 5, we argued that a form of cross validation, in this case k-fold cross validation, is necessary to check for the variation among the model instances and validation sets. Each of the $k$ models can be plotted in ROC space as well. To evaluate the variation among the model instances, we use a performance metric called area under the curve ($AUC$), which is simply the area under the ROC curve. This metric is used because it is a single number metric, which is based on the convex hull of $TPR$ and $FPR$ combinations. It is less useful to compare $TPR$ and $FPR$ combinations, because depending on the operating conditions,

different combinations may be relevant. Figure 20 shows all the ROC plots for each model and the mean, standard deviation and 95% confidence interval of the $AUC$ metric are included.

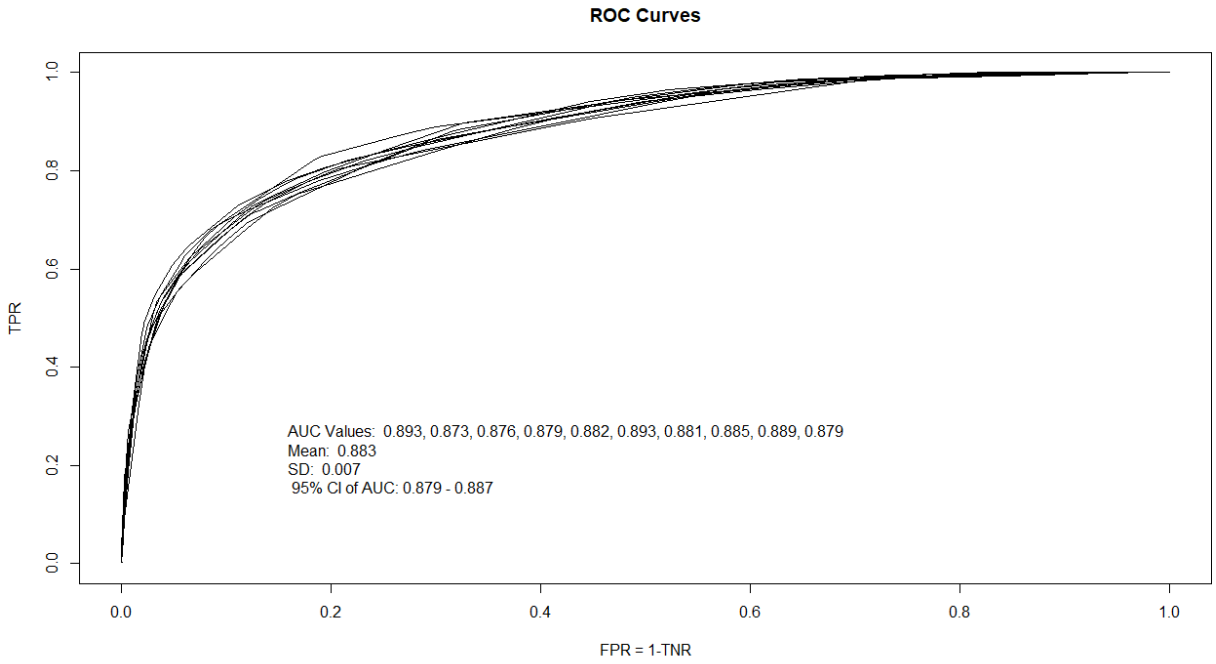



*Figure 20 - ROC curves for each model instance resulting from k-fold cross validation.*

Judging from Figure 20, the model instances are fairly alike, indicating that the model performance is consistent over different sets of training and evaluation data. However, the variation between the curves seems to be larger around the middle of the curves. In the case that operating conditions are chosen such that the middle of the curves are of interest, then variation between the model instances is larger. In addition, the confidence interval is only indicative, given that it assumes the $AUC$ values to be independent and normally distributed, which is probably not true. Therefore, a visual interpretation might be more appropriate. Finally, it should be noted that $AUC = 0.5$ is basically the lower bound on the model performance, because any model performing worse, could be improved by simply turning around the decision (see Section 5.7).

## 6.2. Model Hyperparameter Selection

Almost all learning algorithms have a set of hyperparameters, which can be tuned to stimulate desired model behavior. Random forests are no exception. In this research, three modeling parameters are examined in more detail: *number of trees* ($nTree$) to construct a random forest, *number of randomly sampled features* ($mTry$) at each node in a decision tree and the *sample size* ($ss^+, ss^-$) indicating the number of randomly sampled data instances to construct each tree, specified per class label.

### 6.2.1. Number of Trees

In general, one cannot add too many trees to a random forest in the sense that the average error should not be increasing as more trees are added. However, adding more trees takes up more computation time and at a certain point, adding more trees will not result in a lower error rate. Section 5.6.2 introduced the concept of *out-of-bag* (OBB) error estimation, which provides a convenient way to determine the error

rate of a model. The number of trees and the resulting error rate can be plotted to obtain a graph that indicates at what number of trees the error rate stabilizes. Figure 21 shows such a graph for a model instance of the basic model defined in section 6.1. The OOB error estimation per class is shown as well as the average over all cases.

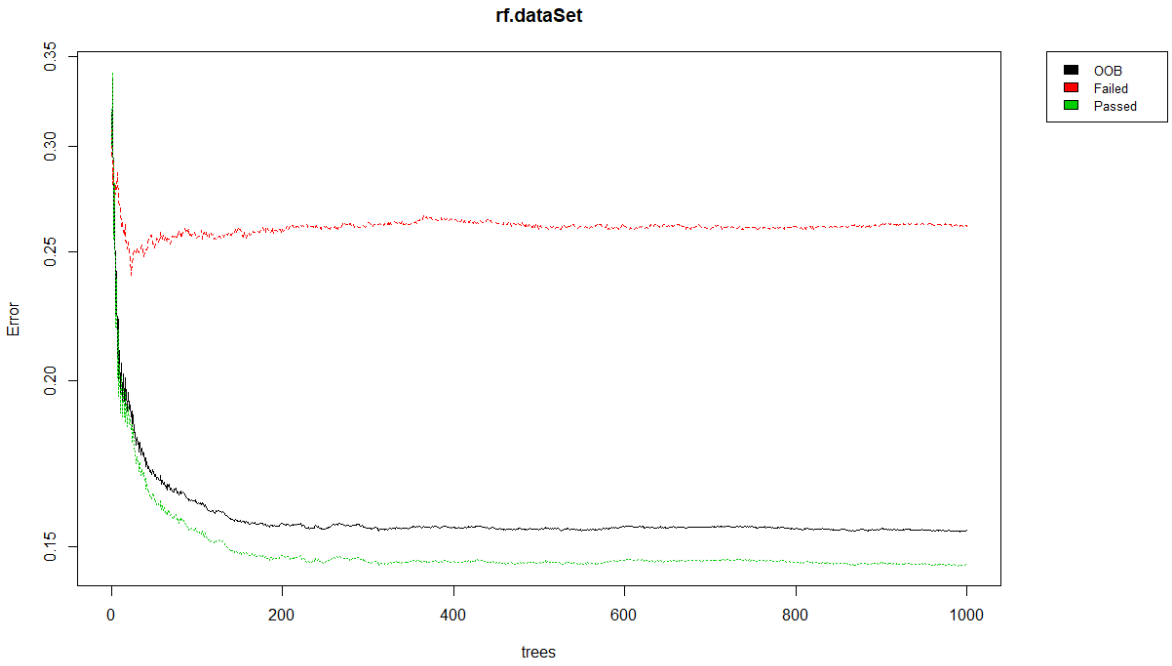



*Figure 21 - OOB Error Estimation for a model instance of the basic model.*

As can be deduced from Figure 21, the error rate seems to stabilize at roughly 300 trees. The number of trees used in the basic model is 1000, which suggests the chosen default value for the number of trees in the basic model is adequate. Note that this is the error rate obtained when using majority voting, i.e. when the class label is chosen with the most votes. This would be analogous to only one point in ROC space. However, when using k-fold cross validation we can compare whole ROC curves using the $AUC$ performance metric. To compare our findings, we build several models with different values for the number of trees. For each model k-fold cross validation is used to determine the variation among the model instances. Table 17 shows the results and they are visually represented in Figure 22.

*Table 17 - Results for models based on different values for nTree including the basic model (gray row).*

| $nTree$ | $mean\ AUC$ | $sd\ AUC$ |
|---|---|---|
| 100 | 0.877 | 0.012 |
| 500 | 0.882 | 0.012 |
| 1000 | 0.883 | 0.009 |
| 1500 | 0.883 | 0.008 |
| 2000 | 0.883 | 0.008 |

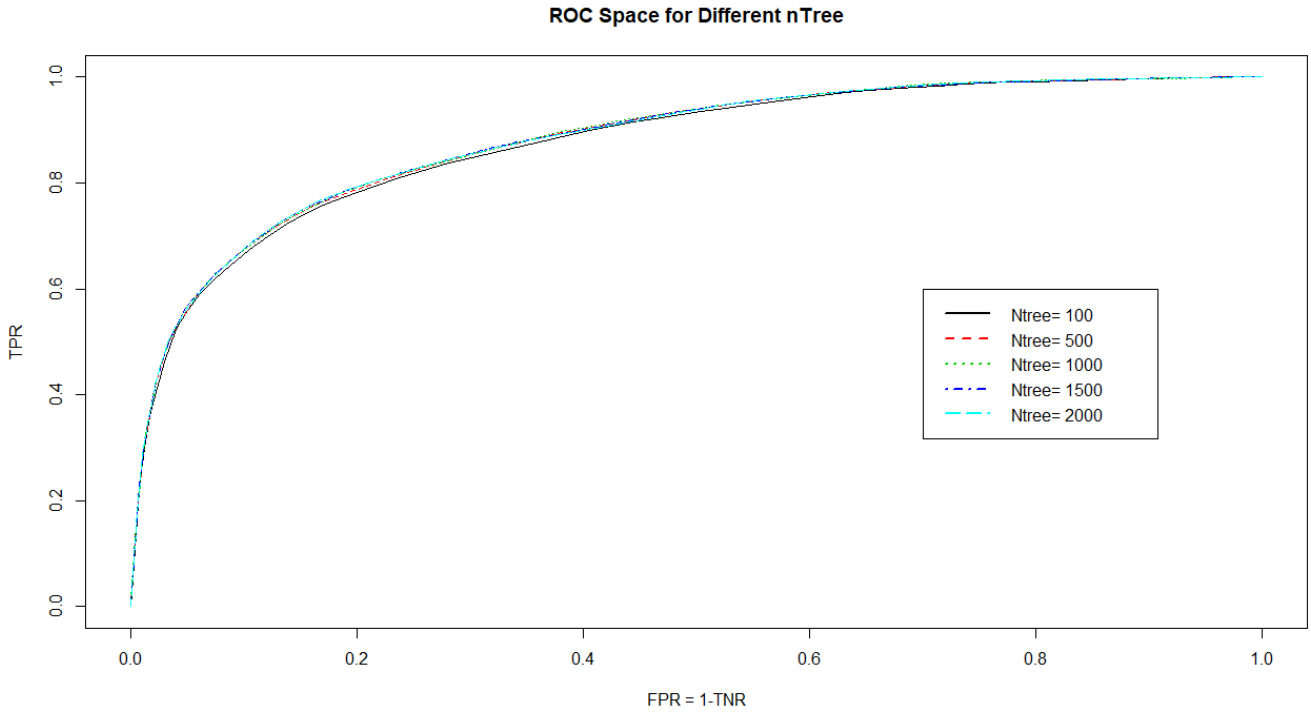



*Figure 22 - Average model performance for different values of nTree, suggesting the model is insensitive to changes for nTree.*

Judging from Table 17 and Figure 22 it seems that the number of trees does not really matter in terms of average model performance or the variation between the model instances. This can be explained by the fact that the error rate for failed performance tests already stabilized around 100 trees. While determining the $AUC$ in ROC space, $TPR$ and $FPR$ are given equal importance, thus masking the slightly higher error rate of the negative class labels at 100 trees.

### 6.2.2. Number of Randomly Sampled Features

The number of randomly sampled features at each node in the construction of a decision tree, referred to as *mTry*, is another important hyperparameter. James et al. (2013) explain that choosing *mTry* too low may result in the possibility that certain strong predictors are infrequently used in trees, which means that their predictive value may not be used to full extent. On the other hand, choosing *mTry* too high might lead to decision trees being too similar. Consequently, the predictions from each individual tree may be highly correlated. Unfortunately, averaging many highly correlated quantities does not lead to as large a reduction in variance as averaging many less correlated quantities. Random forests try to overcome this problem by only allowing to review a subset of features at each node in the decision tree to create lower correlated trees. It is common practice to use $mTry = \sqrt{p}$, where $p$ equals the number of features present in the dataset. The basic model uses $mTry = \sqrt{p}$ as well ($mTry = \sqrt{63} \approx 8$). Even though it is recommended to do this, we still test whether other values for $mTry$ lead to better results. Table 18 shows the different values of $mTry$ that have been used along with the standard deviation of the area under the curve and the mean $AUC$. Figure 23 represents a visual representation of the ROC curves that result from each value of $mTry$, but recall that these are the average performance over all

model instances generated by k-fold cross validation and thus the variation among the individual model instances is not shown.

*Table 18 - Results for models based on different values for mTry including the basic model (gray row).*

| $mTry$ | $mean\ AUC$ | $sd\ AUC$ |
|---|---|---|
| 1 | 0.842 | 0.01 |
| 4 | 0.878 | 0.008 |
| 8 | 0.883 | 0.009 |
| 16 | 0.885 | 0.001 |
| 32 | 0.885 | 0.007 |
| 64 | 0.884 | 0.007 |

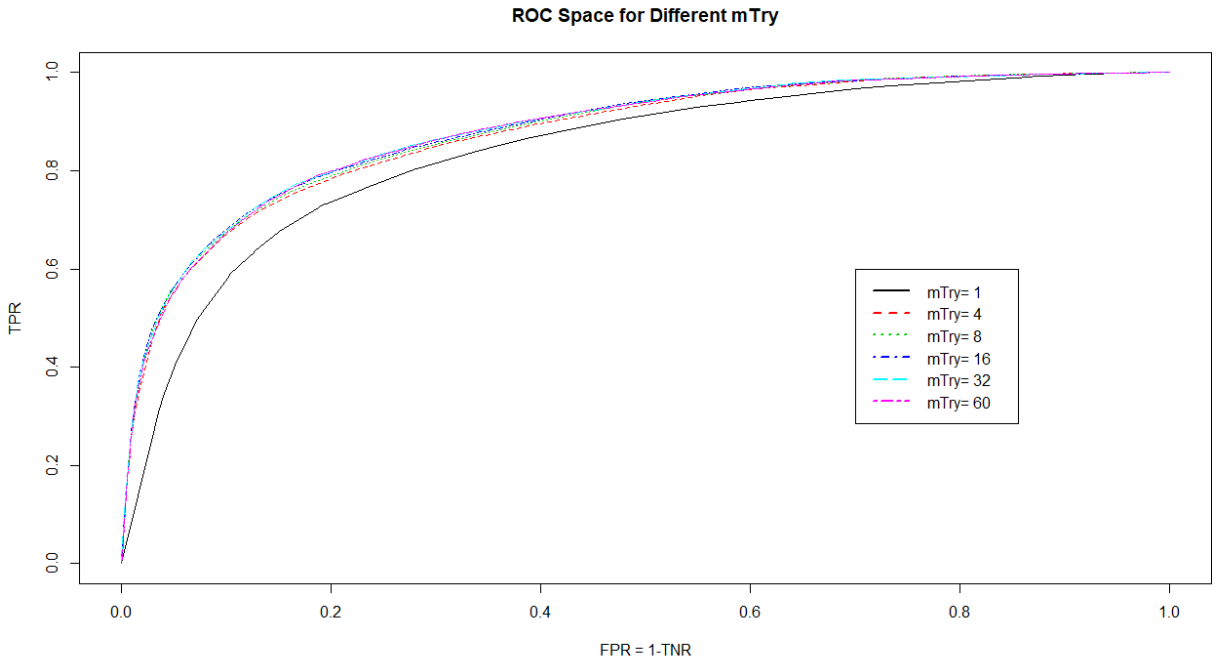



*Figure 23 - Average model performance for different values of mTry.*

Judging from both Table 18 and Figure 23, the value of $mTry$ does seem to influence the average model performance. However, as soon as it is large enough, say $mTry \geq 8$, the model performance seems stable, suggesting $mTry = 8$ is an appropriate decision for the this hyperparameter. In addition, similar performance for lower values of $mTry$ suggest that features are correlated, because apparently similar predictive value can be obtained with fewer features.

### 6.2.3. Sample Size

The hyperparameter $sample\ size$ influences the way samples are randomly drawn from the training dataset to train the random forest classifier. By default, each tree is built by randomly drawing $n$ samples from the training set with replacement, where $n$ is the number of observations in the training set. Without intervention, the class skew in the randomly drawn samples will be similar to that of the used training set.

In many cases, this may be appropriate, but in our case that would lead to a focus on the negative samples, as they are by far the largest fraction of the samples. This phenomenon is also explained in Section 5.4. By intentionally fixing the number of randomly sampled positives and randomly sampled negatives, it can be ensured that a balanced dataset is used to train the random forest. This stimulates equal importance of classifying positives and negatives. If majority voting is used, then this hyperparameter is extremely important, because it is would be the only way to influence the bias of the model towards either positives or negatives. However, in Section 5.7, the threshold value $T$ has been introduced which can be used to influence this bias as well. So, the bias of the model can be influenced by both $T$ and $sample\ size$. Since, $T$ is determined based on the operating conditions after the model is trained, we can experiment with different values of $sample\ size$, that is, different fractions of positives in the training set, denoted with $p^+$. Table 19 shows the different sample sizes that have been tested. Note that the sample sizes are relatively small compared to the whole dataset, because the initial dataset in this research was also smaller. The *randomForest* (Liaw & Wiener, 2002) package used to build this model does not allow larger sample sizes per class label than the number of cases present in the training set and therefore we under-sample the negative cases. There are several different popular under-sampling methods available (Liao, 2008). Since, random forests use sampling with replacement it is most intuitive to use Bootstrap Under-sampling, which is essentially the same as the default sampling methods for random forests, but then with less samples. Several models have been built with larger sample sizes as well, but that did not impact the results.

*Table 19 - Results for models based on different values for $ss^+$ and $ss^-$ including the basic model (gray row).*

| $ss^+$ | $ss^-$ | $p^+$ | mean AUC | sd AUC |
|---|---|---|---|---|
| 90 | 810 | 0.10 | 0.876 | 0.008 |
| 225 | 675 | 0.25 | 0.882 | 0.009 |
| *450* | *450* | *0.50* | *0.883* | *0.009* |
| 675 | 225 | 0.75 | 0.875 | 0.012 |
| 810 | 90 | 0.90 | 0.859 | 0.009 |

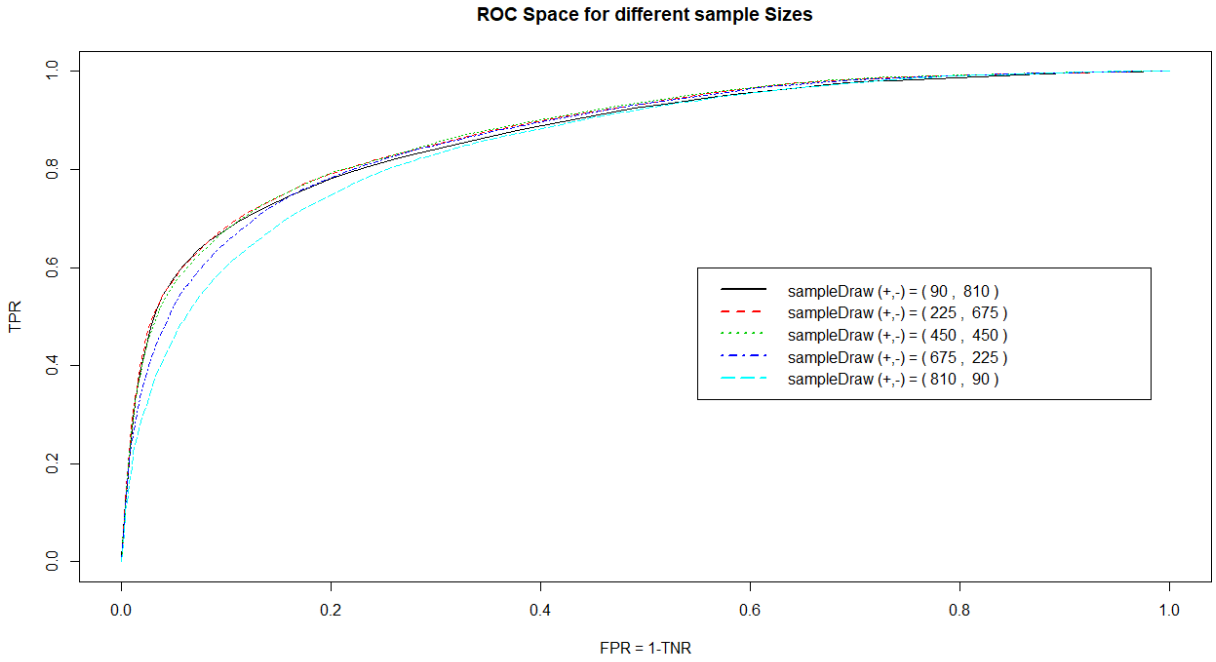



*Figure 24 - Average model performance for different values of $ss^+$ and $ss^-$.*

Judging from Table 19 and Figure 24, the balanced case does give the best average performance. However, only the highly imbalanced datasets skewed towards the positive class seem to give notably inferior results. Therefore, we can safely assume that a decision for a balanced dataset in the basic model is appropriate, but it does not yield significantly better results than using a negatively skewed, i.e. $p^+ = 0.1$ training set, which is a similar to the original dataset. Introducing $T$ seems to make stratification for this example superfluous.

## 6.3. Feature Selection

In many classification algorithms, feature selection is an important topic, because it affects what data can be used to train a classifier. Usually, features have a different importance with respect to predictive power of the classification model. In some cases, using features that add little to no predictive power can even reduce the overall performance of the model. However, a random forest algorithm is quite robust to this problem, because it has the option to choose between several features at each node in the decision tree. Nevertheless, it is good practice to verify if that also holds in our case, because in the basic model, we argue to use all available features. To test if this is an adequate decision we compare the performance of different model implementations using top-$n$ of features for different values of $n$. An additional reason to test how different sets of features affect the model performance is to gain insight in how powerful it is if less data is available. This can be beneficial, because that could perhaps increase the number of systems that the model could be applied to, because not all systems have complete sets of data as discussed in Section 4.2.4.

To check the performance of the model with different feature sets we use the following approach. First, we determine the importance of the features by ordering them according to their decrease in Gini impurity, which can be obtained by running the basic model including all features. The Mean decrease of the Gini impurity is basically the average decrease of the node impurity that results from using a certain

feature in a decision tree. Then the model is trained using only a subset of the most important features. The subset is decreased until poor results are obtained, or few features remain. Table 20 shows different subsets of features that have been used and their associated results. In addition, the average performance is visually represented in Figure 25 by plotting them in ROC Space.

*Table 20 - Different feature sets and their model performance including the basic model (gray row).*

| Number of features | $mTry$ | $mean\ AUC$ | $sd\ AUC$ |
|---|---|---|---|
| 63 (all, basic model) | $\sqrt{63} \approx 8$ | 0.883 | 0.009 |
| Top 30 | $\sqrt{30} \approx 5$ | 0.882 | 0.008 |
| Top 20 | $\sqrt{20} \approx 4$ | 0.885 | 0.006 |
| Top 10 | $\sqrt{10} \approx 3$ | 0.879 | 0.008 |
| Top 5 | $\sqrt{5} \approx 2$ | 0.852 | 0.01 |
| Top 3 | $\sqrt{5} \approx 2$ | 0.84 | 0.006 |
| Top 2 | $\sqrt{2} \approx 1$ | 0.792 | 0.01 |

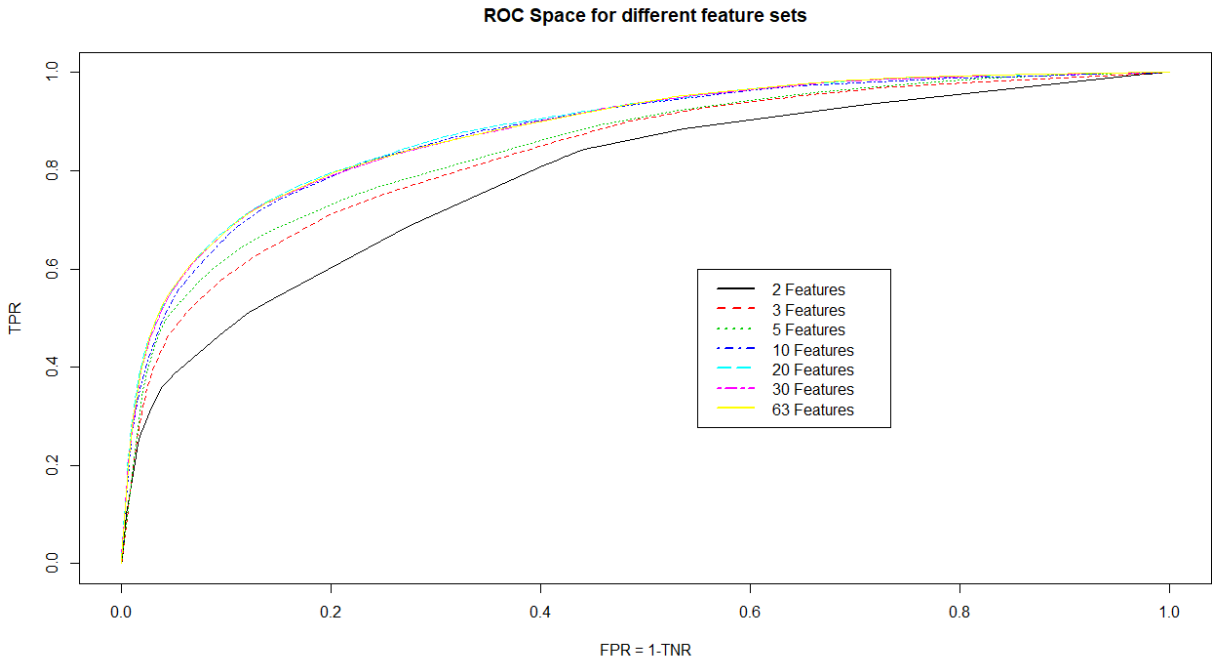



*Figure 25 - Different Feature sets and the resulting ROC curves.*

From Figure 25 it becomes clear that using top-$n$ featres for $n \geq 5$ leads to similar performance. However, when $n < 5$, the performance of the model drops noticeably. This suggests that the majority of the features are relatively useless. This is somewhat in line with the expectations, because in Section 4.3 we argued that the majority of the quality assurance measurements rarely impact the monitor performance outcome. Therefore, these values are likely to be of no value.

A closer look at what type of features are most important reveals that the prediction is mainly based on previous test values relating to the contrast of the monitor, as well as the country in which the devices are located. Other types of features such as age-based and usage-based features have only a small impact on the results, because they are not part of the top 10 features.

## 6.4. Dataset Selection

In any datamining project decisions about data selection are vital to how the model behaves as well as how the model performance should be interpreted. This section examines an interesting observation discussed in Section 4.3 where we showed that roughly 58% of the failed tests are consecutive in nature. A performance test is considered a consecutive failure if the performance test yields a failed outcome and the most recent performance test (present in the cleaned dataset) on the same monitor also yielded a failed outcome. Since previous test results are available for prediction, it can be argued that this makes it relatively easy for the model to identify failures. If the proportion of consecutive failures in the dataset is large, then the question may be posed if the model is actually performing the way it should. In other words, is the model actually identifying failures, or is it just identifying failures that are consecutive? We emphasize that for the operational context to which the model is to be applied it does not matter, since it is a representation of the real world. Maintenance actions are not different for consecutive failures, so from a practical perspective it does not make sense to exclude those from the dataset. However, it is valuable to gain insight into why the model behaves as it does and how this phenomenon affects the model performance.

To gain insight into this phenomenon, the prediction scores $s(x)$ of non-consecutive failed performance tests are compared to consecutive failed performance tests as well as the tests with a passed outcome. Figure 26 shows this comparison.

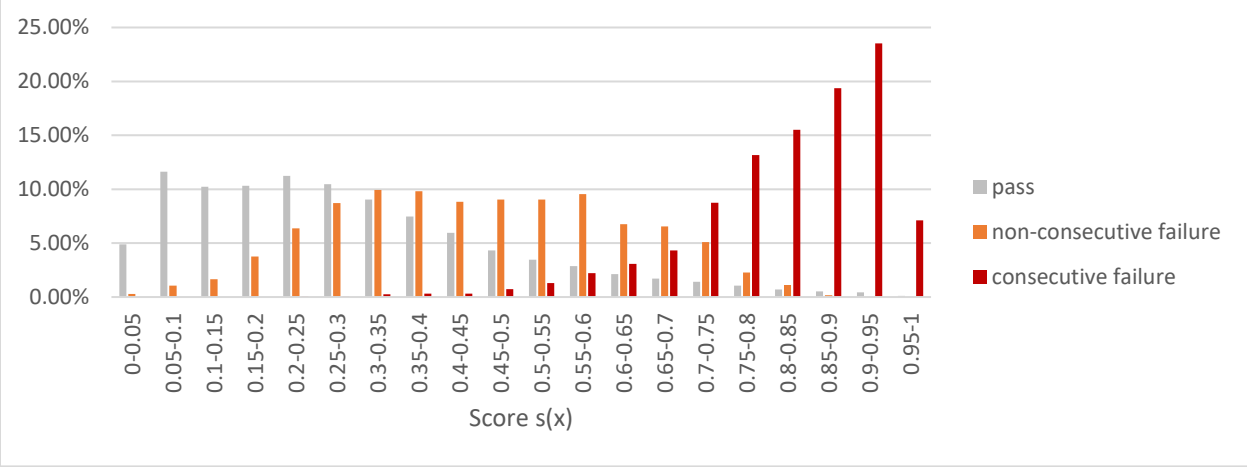



*Figure 26 - Score distribution for consecutive failures (orange) and non-consecutive failures (blue).*

Figure 26 clearly demonstrates that scores for consecutive failures are noticeably higher than scores for non-consecutive failures. This also explains why good trade-offs can be reached with High $TPR$ rates, because a significant part of the failures has good scores.

Given that $s(x)$ is generally higher for consecutive failures, it is interesting to observe how the model would perform if consecutive failures are excluded. Figure 27 shows the comparison of the basic model applied to both the dataset with consecutive failures and without consecutive failures.

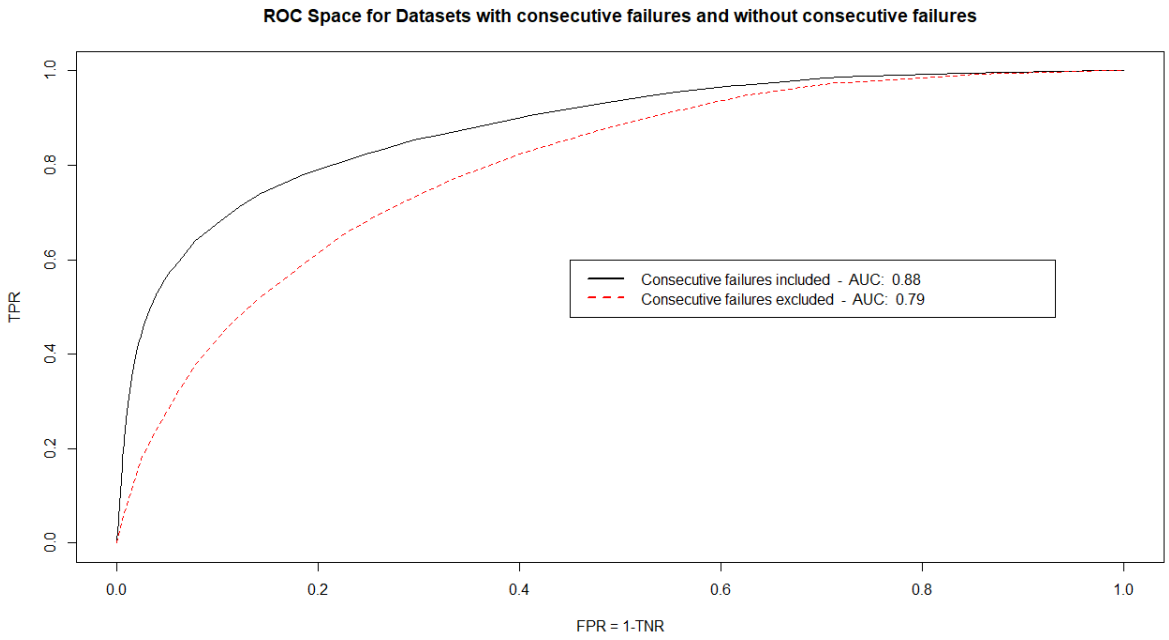



*Figure 27 - ROC curves for the basic model applied to a dataset with consecutive failures and without consecutive failures.*

Figure 27 clearly shows that the performance of the model on the dataset without consecutive failures performs worse. In addition to the earlier made argument, there is another reason this may hurt performance. The already skewed dataset becomes even more skewed, because roughly 58% of the failures are excluded from the dataset. Therefore, the fraction $p(+)$ of positives is roughly $3.5\%$.

Considering both Figure 26 and Figure 27, it is definitely the case that consecutive failures are largely responsible for the performance of the model. This is to be considered when interpreting the results, but as argued before, it remains a justifiable approach for the practical problem the model is to be applied to. Hence, the consecutive failures are included in the basic model.

## 6.5. Two-Stage Learning Algorithm

During the development of the decision support model in Chapter 5, an alternative way of developing the binary classifier has been considered: a two-stage learning algorithm. Instead of directly predicting the outcome of a performance test using all previous quality assurance measurements, the outcome of each subtest is predicted first. Then, the predictions of the subtests could be used as a new feature set to make the final prediction about the performance test. Previous data modeling efforts showed that higher predictive performance could be obtained for several of the individual subtests. Since the goal is to predict on the level of performance tests, the individual predictions should then be aggregated to form a prediction about the performance as a whole. The idea behind this concept was that aggregating better individual predictions could lead to a superior performance compared to the basic model.

The two-stage learning algorithm operates in two stages. In the first stage, a random forest model is constructed for each included subtest. Only 9 of the 17 subtests have been used, because some subtests have considerably skewed class distribution ($p(+) \leq 0.01$). Not only does this affect their respective performance, it also indicates that they rarely ever affect the outcome of the performance test as a whole, because they rarely fail. Therefore, these subtests are omitted. In the end, only the contrast and vignetting

subtests are included. In the first stage, a dataset is created consisting of nine features and the original class labels. The feature values represent the individual predictions of the included subtests. Then, using k-fold cross validation the same number of data instances can be generated as the size of the original dataset. Then, the second stage uses the dataset that results from the first stage to construct a new random forest classifier. This random forest algorithm generates the final prediction in the form of a score. Figure 28 visualizes this process.

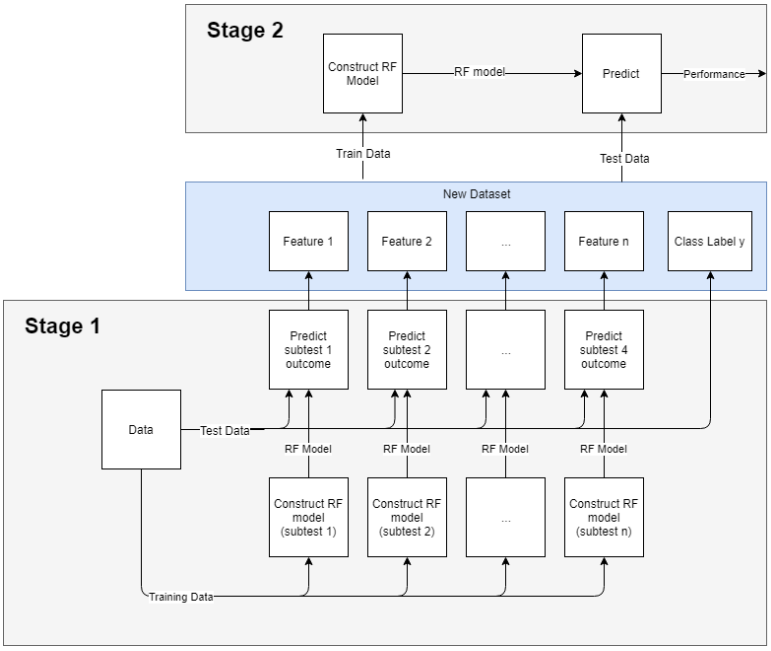



*Figure 28 - Concept of the two-stage learning algorithm using random forests*

During the evaluation of the results, it showed that the basic model and the two-stage learning algorithm showed similar results. In hindsight this is to be expected, since no new data is introduced.

Given that this is the case, the basic model is favorable, because it is less complex and is significantly faster. The two-stage learning algorithm must generate 10 instances of random forest classifiers, compared to one in the basic model. So roughly speaking it is 10 times slower. It is unknown if the two-stage learning algorithm would be more appropriate for other performance tests, but for the monitor performance test it does not add any additional value.

## 6.6. Model Validation with Alternative Validation Dataset

Whenever machine-learning algorithms are involved, careful attention should be paid to overfitting. Overfitting is a modeling error that occurs if the learning algorithm captures noise in data. Then, the algorithm is too perfect for a particular dataset and consequently the model is less appropriate for other, but similar datasets. Since this chapter has introduced a set of actions to optimize the performance with respect to a given dataset, there is risk of overfitting. To check that this model will perform equally well on future data, a new dataset is constructed consisting of data generated after the initial dataset was constructed. The new validation dataset only consists of performance test carried out after July 2018 and has a size of approximately 2500 cases. We note that this is significantly less than the $\pm$ 56,000 samples used to present the basic model.

To see whether using the new validation set gives similar results as using the k-fold cross validation on the original dataset, the model in both cases is trained using the original dataset. Then, two outcomes are generated. First, the model performance is evaluated using the k-fold cross validation technique, similar to the way performance of the model is determined through the rest of Chapter 6. Then in the alternative method, the validation set is applied to the same model instances generated during the k-fold cross validation. The outcome reflects the performance of the basic model under new data. The comparison is shown in Figure 29.

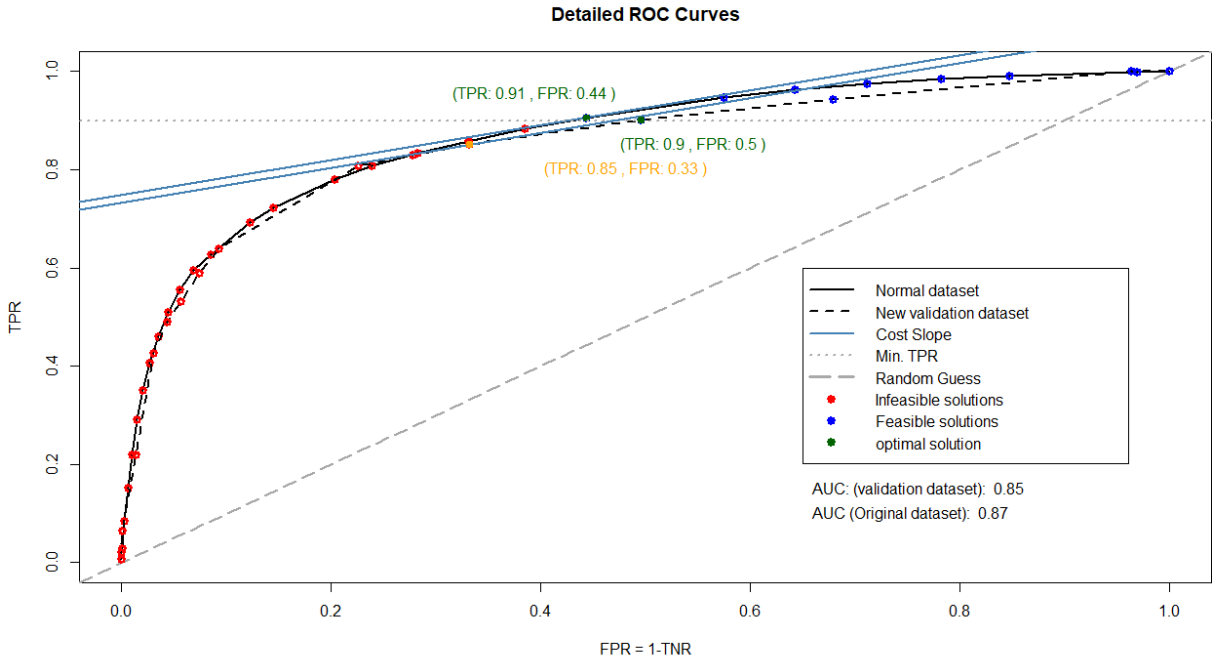



*Figure 29 - Detailed ROC curves for the basic model tested with the original dataset and with the new validation dataset.*

Figure 29 shows that the difference in performance over the whole curve is quite small. This is a sign that the model is likely to perform similar on future data compared to the results shown throughout Chapter 6. We note that even though the difference between the two curves in terms of $AUC$ is small, the optimal points on the curve do differ notably for the arbitrary, but realistic cost slope. For example, the outcome using k-fold cross validation with the normal dataset shows an optimal trade-off point of $TPR = 0.91$ and $FPR = 0.44$ compared to an outcome of $TPR = 0.90$ and $FPR = 0.50$ using the validation dataset. This translates to approximately 5% less tests saved with similar $TPR$ rates, suggesting that the performance presented throughout Chapter 6 may be slightly overconfident. Again, the cost slope drawn in this graph is realistic, but arbitrary, since operating conditions are determined at a later stage when the model is to be applied. Note that the trade-off between $TPR$ and $FPR$ becomes more extreme for higher values of $TPR$. Therefore, the difference in performance between the two curves in Figure 29 is more extreme for higher values of $TPR$ as well. For example, for $TPR = 0.97$, the difference in $FPR$ is approximately 0.15. Even though performance seems similar, it largely depends on how the operating conditions are chosen. Figure 30 highlights this relation.

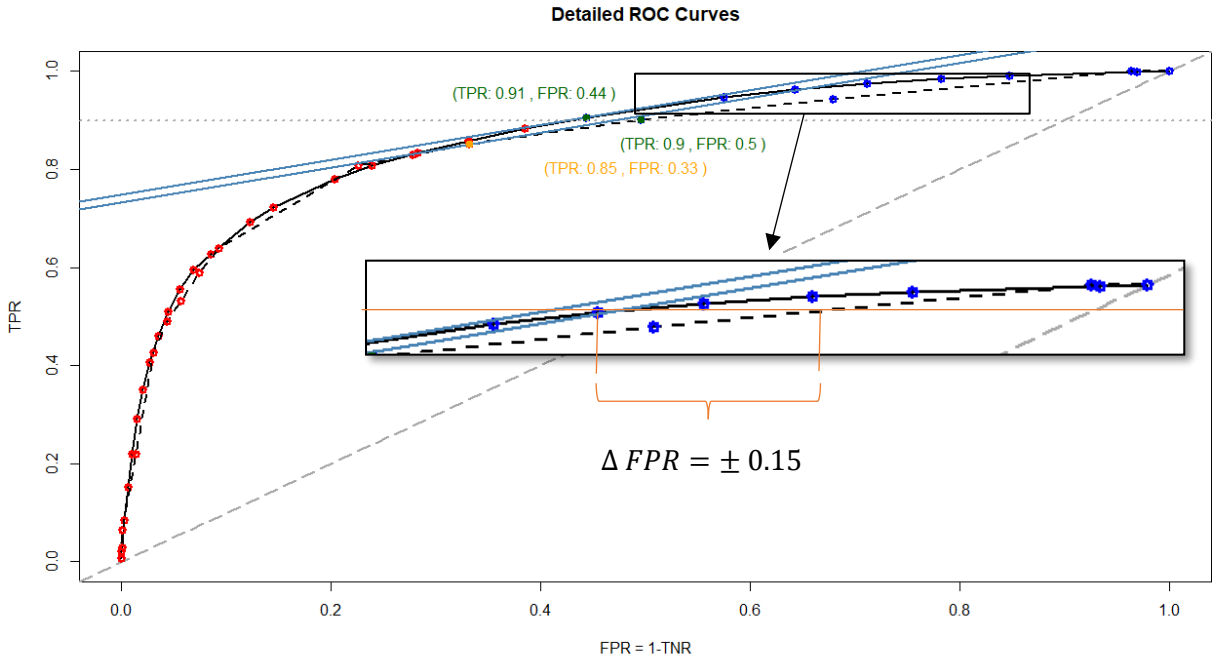



*Figure 30 - Example of large FPR delta for a given TPR between two test outcomes.*

However, Figure 30 does not fully capture the performance of the future dataset. Recall that threshold $T$ is determined such that the most optimal trade-off is selected based on the original dataset. When the model is applied in practice, $TPR$ and $FPR$ cannot be determined, because TN and FN are unknown (if a performance test is skipped, there is no way to verify if it was correct course of action). Figure 30 only shows the possible trade-offs that can be obtained, but not for which values of $T$. In other words, when using the model in practice, $T$ is already determined, so it is more relevant to check if the similar $TPR$ and $FPR$ are obtained for $T$. Therefore, the performance of both $TPR$ and $FPR$ as a function of $T$ for both the original dataset and the new validation set are presented in Figure 31. The graph contains 10 curves for each dataset, because when using k-fold cross validation, k models are constructed. To gain insight in the consistency of the model, all curves are presented.

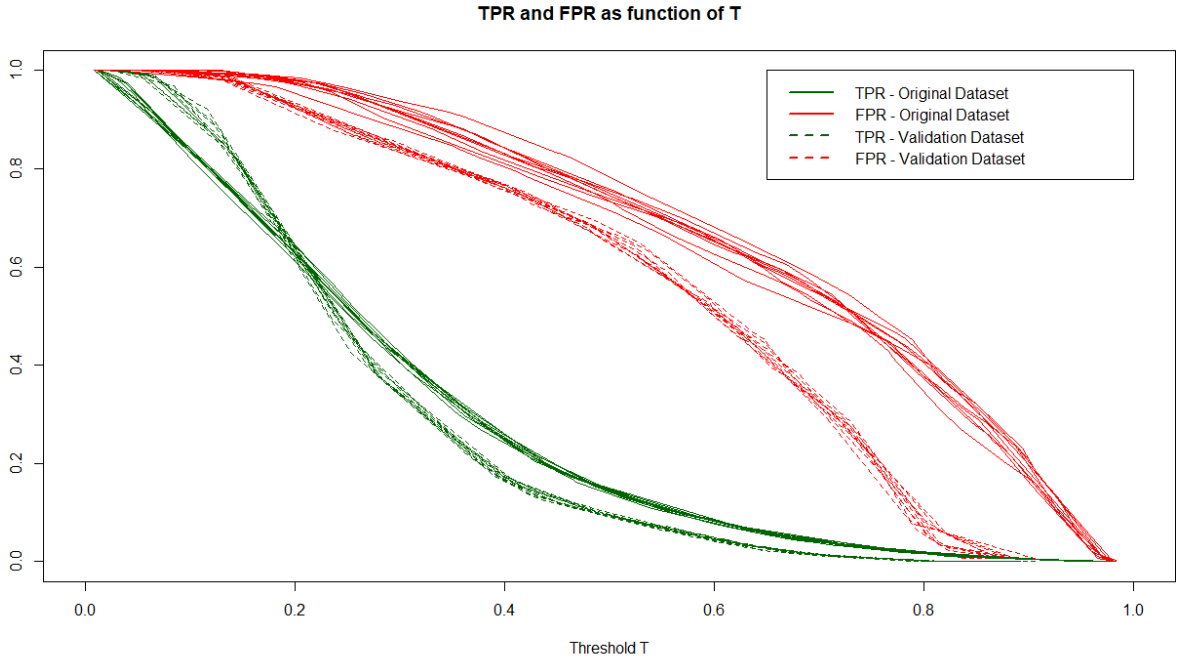



*Figure 31 - TPR and FPR for the original dataset and the validation dataset as a function of T.*

From Figure 31 it can be observed that for $T \leq 0.2$, the model performed even better on the validation set in terms of $TPR$ and slightly worse in terms of $FPR$. However, for $T > 0.2$ the performance on the validation set is considerably worse. In this project, high values of $TPR$ are more important than lower values of $FPR$. Consequently, only values for $T \leq 0.2$ are likely to be relevant. Fortunately, this is the region where results are for both the original and validation dataset seem promising. In addition, for $T \leq 0.2$ the trade-off between $TPR$ and $FPR$ is quite similar, but for $T > 0.2$, the difference between the dataset in terms of the $TPR$ and $FPR$ trade-off is getting bigger. A difference in trade-off would mean that the model would not behave optimally with respect to the operating conditions.

# 7. Conclusions

The goal of this research is to reduce planned maintenance by developing a decision support model that is able generate recommendations for the decision to skip or conduct planned maintenance. This chapter starts with summarizing the main findings and remarks. Then, in Section 7.2 the potential impact of the proposed decision support model is discussed.

## 7.1. Main Findings

To achieve the goal for this research project, the DM-CRISP methodology has been used. In this research, all the phases have been addressed. Key findings in each phase are discussed shortly.

### 7.1.1. Data Understanding & Data Preparation

The data understanding and data preparation phase have taken the majority of the research time. During these phases, it became clear that large amounts of data are available. Three categories of data are examined more closely: *device and maintenance data*, *system log data* and *performance test data*. Often, extracted data (using SQL) is not directly ready for use. Chapter 4 is dedicated to the process of how data is extracted and prepared for the modeling phase. It discusses several decisions regarding the data preparation. A java application is written that reads, cleans and adjusts data to these decisions accordingly. After the application processed the data, an overview is generated containing information about the final dataset as well as the effect of the data preparation decisions on the dataset. It showed that approximately half of the data was excluded and that the class skew in the final dataset was large ($p(+) = 0.08$ and $p(-) = 0.92$). Since a significant part of the data is excluded, we noted that the scope to which the model can be applied with the expectation of similar results presented throughout this research is implicitly reduced as well.

### 7.1.2. The Decision Support Model

The decision support model is based on two parts. The first part consists of a binary classifier. We noted that in essence any binary classification algorithm can be used that can express a probability estimate per class label. However, we argued that in particular random forests algorithms are a good decision for the presented case study in Chapter 6. The second part is a sophisticated decision rule that considers the output of the classifier as well as the operational context. It is important to note that the inputs for the decision rule can be changed after the binary classifier is constructed. This allows us to use the same binary classifier in different operational contexts. The operational context is characterized by both operating conditions and operating constraints. Operating conditions represent the class skew and the cost skew, while operating constraints reflect certain safety-related requirements. In addition, we explained that the decision support model as a whole can be considered cost sensitive, but we noted that the learning algorithm in itself is not cost-sensitive. This is beneficial, because operating conditions can change over time and that does not require the model to be retrained.

We argued that the performance of classifiers can be evaluated by different performance metrics but concluded that using sensitivity and specificity is most appropriate for this research problem. In addition, we showed that the performance of the model could be evaluated by plotting the performance in ROC space and that the ROC curve is essentially an overview of all the trade-offs between $TPR$ and $TNR$. The operating conditions are combined in a single parameter referred to as the cost slope and it is used to determine the optimal point on the ROC curve. The operating requirements simply limit the region on the ROC curve that is feasible with respect to the safety related aspects.

### 7.1.3. Case Study: Monitor Performance Test

Chapter 6 described a case study where the decision support model is applied to the monitor performance test. The basic model shows promising results, but it is clear that the operating conditions and constraints severely affect the value of the model. Especially in the regions on the ROC curve with either very low or high values for $TPR$ and $FPR$, trade-offs become extreme. For example, increasing $TPR$ from 0.95 to 0.96 results in a far larger increase in $FPR$ than increasing $TPR$ from 0.80 to 0.81. Experiments with different values for hyperparameters showed that the model performed stable except for extreme values for the hyperparameters. Models with different feature sets showed that only a small group of features is important and that excluding the less important features has little to no effect. The most important features are previous contrast measurements and the country the device is installed in.

Then, the model is tested on the same dataset, but with the exclusion of consecutive failures (both this and the most recent performance test on the same monitor have failed). Results show that the classification model is far better at classifying consecutive failures, suggesting that the consecutive failures are the primary reason that the model performs well. However, since maintenance actions are the same for consecutive and non-consecutive failures it does not matter from a practical perspective. Regardless, it is valuable to know why the model performs well.

In an attempt to gain higher predictive performance, the basic model is adjusted to a two-stage learning model. In this model, subtests are predicted first, and based on these predictions the final outcome is predicted. The idea behind this concept was that aggregating better individual predictions could lead to a superior performance compared to the basic model. Results showed that the performance was similar to that of the basic model. Since the basic model is more transparent and faster, this model is preferred.

Finally, the basic model has been validated with an alternative dataset containing performance tests that were generated after the initial construction of the original dataset discussed in Chapter 4. This has been done to see if the basic model suffered from overfitting. The validation set shows a slightly worse performance which is to be expected. Therefore, Chapter 6 might represent slightly overconfident results, so careful interpretation is required regarding the model performance. This comparison also confirmed that the performance of the model is highly dependent on how the operating conditions are set, because difference in performance is notably larger in regions with low $TPR$ and $FPR$.

## 7.2. Model Value

This section provides a discussion on the potential impact of a successful implementation of the proposed decision support model. Since we argued that the model value and model performance strongly depend on the way operating conditions are set, we define three scenarios: a strict, medium and lenient scenario with different requirement on the $TPR$. Alternatively, we could have used three different values for the cost slope, but a minimum $TPR$ more clearly illustrates the different scenarios. To get insight into the impact of the model, we consider the yearly impact in terms of tests saved and the number of undetected faulty systems.

Of the total number of IXR devices that are served by Philips, more than half are used in the development of the proposed decision support model. We acknowledge that some of the devices in the dataset may not be operational anymore, but similarly it could be argued that this model could be applied on more devices than just the devices present in the dataset. Regardless, $M$ is used to represent the number of applicable devices. The monitor performance test is performed bi-annually. Therefore, the number of

yearly performance tests, $N$, can be expressed with $N = 2 \cdot M$. Finally, we observe from historic data that $p(+) = 0.08$ and $p(-) = 0.92$, meaning that 8% of the performance test had a failed outcome and 92% has a passed outcome. Note these numbers are obtained from the cleaned dataset and by using these numbers it is implicitly assumed that future class skew will be the same. Then $E[+]$ and $E[-]$ can be determined:

$$E[+] = p(+) * N$$

$$E[-] = p(-) * N$$

Recall that depending on the operating conditions and constraints different combinations of $TPR$ and $FPR$ are optimal. So, for each scenario a different trade-off between $TPR$ and $FPR$ can be determined. The three scenarios are shown in Figure 32.

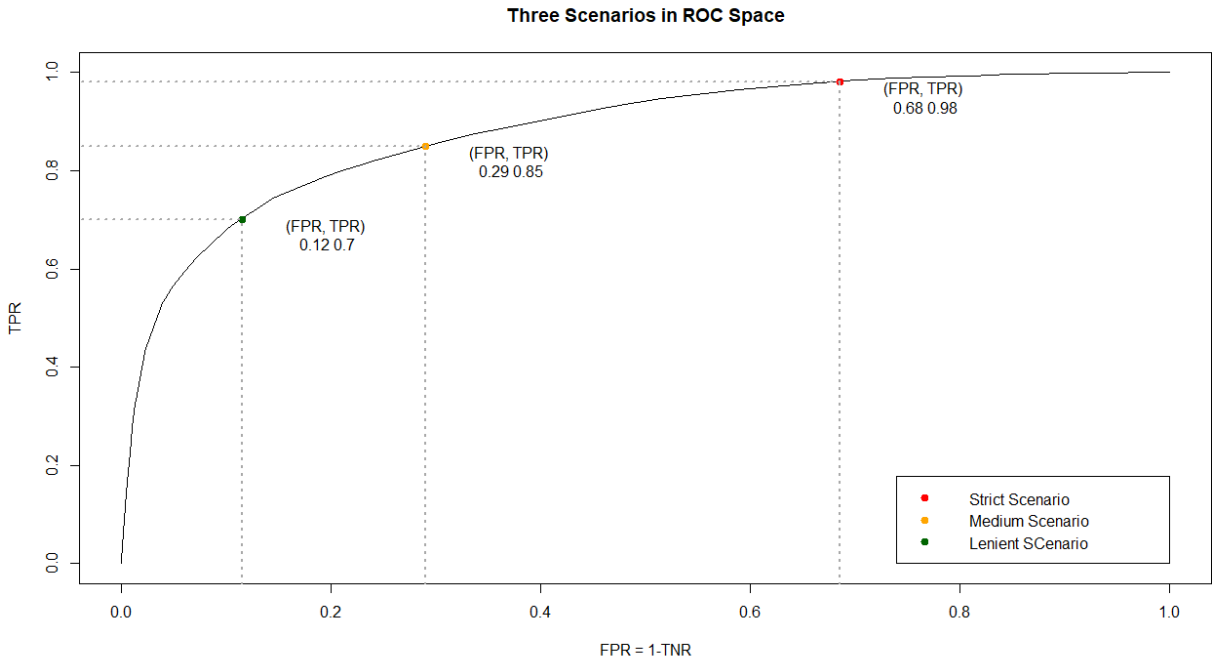



*Figure 32 - Three scenarios in ROC space.*

Then, for each scenario the number of tests saved, and number of misidentified systems can be determined by using the following equations. The results are summarized in Table 21.

$$\frac{E[tests\ performed]}{N} = \frac{FPR * E[-] + TPR * E[+]}{N}$$

$$\frac{E[tests\ saved]}{N} = 1 - \frac{E[tests\ performed]}{N}$$

$$\frac{E[Misidentified\ faulty\ system]}{N} = \frac{E[FN]}{N} = \frac{(1 - TPR) * E[+]}{N}$$

*Table 21 - Naive practical performance example as a percentage of N (number of yearly performance tests)*

| Scenario | $TPR_{min}$ | FPR | % Tests performed | % Test saved % | % FN (misidentified faulty systems |
|---|---|---|---|---|---|
| **Strict** | 0.98 | 0.68 | 70.4 | 29,6 | 0.16 |
| **Medium** | 0.85 | 0.29 | 33.5 | 66.5 | 1.2 |
| **Lenient** | 0.70 | 0.12 | 16.6 | 83,4 | 2.4 |

Table 21 shows that even for the strict scenario approximately 30% of the performance tests could be saved. This suggests that even in the strict scenario significant savings can be obtained. However, we stress that the trade-off between $TPR$ and $FPR$ is extreme. In other words, $0.18 \leq \frac{\Delta \mathrm{TPR}}{\Delta FPR} \leq 0.15$ indicating that a slight decrease in performance may cause a significant increase in $FPR$ when the same $TPR$ is targeted. Section 6.6 showed that a slightly lower performance resulted in a $\Delta FPR = \pm 0.15$. This is a big difference and therefore the strict case should be interpreted with care. In the medium scenario, considerably more tests are saved, but the number of misidentified faulty systems is almost 8 times as large. However, this is a far more robust decision, because the trade-off between both $TPR$ and $FPR$ is not as sensitive, i.e. $0.45 \leq \frac{\Delta \mathrm{TPR}}{\Delta FPR} \leq 0.6$. This means that a slightly lower performance is not as consequential. This can be important when the model is applied in practice, because results may differ. Finally, the lenient case shows that a respectable 83% of all the tests could be saved if a $TPR$ of 0.7 is acceptable. However, it is expensive in terms of the number of misidentified systems. Of course, the best scenario is determined through the operating conditions and constraints, but the medium case seems to be the most balanced.

We acknowledge that one of the operating constraints is implicitly ignored in this example. The constraint ensures that a system is tested at least once within a certain timeframe $\tau_{max}$. This constraint is hard to capture in this example, because an analysis on the device level would be required. We note that for the first year, this constraint is unimportant, because all performance test will meet the constraint anyway. However, for future performance tests the constraint may apply. This constraint has a higher impact on scenarios with lower $TPR$ and $FPR$, because more tests are skipped. Consequentially, performance test will be performed regardless of the model output more often, because they have not been tested in a certain timeframe.

Finally, we argue that the impact of the decision support model is only illustrated for the monitor performance test. However, Philips conducts more performance tests that are similar in nature. At this point, only speculations can be formed as to what performance and value can be obtained when this is also applied to the other performance tests. In any case, the proposed decision support model can be used, given that the available data and structure of the tests are similar in nature.

## 7.3. Contribution to Literature

To the best of my knowledge, there is very little work on predictive maintenance specifically applied in to reduce planned maintenance for the healthcare devices. This research presents a decision support model that is based on combining existing concepts, but its novelty primarily lies in the way it is modified to fit within the healthcare environment. In particular, this work brings the following contribution.

This research proposes a decision support model that is specifically designed to be used for environments like the healthcare industry. These environments are characterized by complex devices that require

maintenance for optimal (clinical) performance as well safety aspects and regulations. Therefore, models that are only driven by costs are not adequate, because implicitly require safety aspects to be expressed as costs. However, the proposed decision support model in this work is extended with two operating constraints which aim to characterize the operational context in terms of safety aspects. These parameters can be used to retain control over the (potential) risk that is associated with the reduction of planned maintenance. As such, the proposed decision support model can be viewed as a cost driven model which is extended with risk limitation parameters.

# 8. Pilot & Implementation

Philips has indicated that the estimated impact of the proposed decision support model is considered interesting enough to continue with the research project. As the next step, a soft pilot is initiated to test the model behavior in practice. A soft pilot refers to the implementation and use of the decision support model, but the output is only documented and not followed. Contrary to a hard pilot (where the outcome of the model is followed) the soft pilot yields more information, because false negatives, i.e. missed hits or undetected system failures, can be observed.

Arguably, the validation described in Section 6.6 could be considered as a soft pilot, because it does not matter if the model is applied before or after the performance test has been conducted. This is true, because the output of the model is not followed anyway. However, a soft pilot is different in several ways, because with a soft pilot more stakeholders are involved. The relevant stakeholders that are responsible for selecting appropriate values for the operating conditions and constraints are forced to start a discussion about what values are appropriate. This is not an easy task, because in many cases, Philips is bound to several safety aspects and other commitments. This will take time and a soft pilot can be executed without the need for a final, explicit answer. In addition, one or several markets will be involved, because markets manage the operational PM activities per region. They will need to be aware of the model and how it should be used.

Finally, we note that the pilot and further implementation of the decision support model is out of scope for this research project. At this point in time, discussions are held to determine future steps.

# Bibliography


Breiman, L. (2001). Random Forests. *Machine Learning*, *45*, 5–32. https://doi.org/10.1023/A:1010933404324

Brown, I., & Mues, C. (2012). Expert Systems with Applications An experimental comparison of classification algorithms for imbalanced credit scoring data sets. *Expert Systems With Applications*, *39*(3), 3446–3453. https://doi.org/10.1016/j.eswa.2011.09.033

Caruana, R., Karampatziakis, N., & Yessenalina, A. (2008). An Empirical Evaluation of Supervised Learning in High Dimensions.

Das, S., Datta, S., & Chaudhuri, B. B. (2018). Handling data irregularities in classification : Foundations , trends , and future challenges. *Pattern Recognition*, *81*, 674–693. https://doi.org/10.1016/j.patcog.2018.03.008

Domingos, P. (1999). MetaCost : A General Method for Making Classi ers Cost-Sensitive. In *In Proceedings of the 15th international conference on knowledge discovery and data mining* (pp. 155–164).

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, *27*(6), 861–874. https://doi.org/10.1016/j.patrec.2005.10.010

Garcia-Lacencina, P. J., Sanch-Gomez, J. L., & Figueiras-Vidal, A. R. (2010). Pattern classification with missing data : a review. *Neural Comput & Applic*, *19*, 263–282. https://doi.org/10.1007/s00521-009-0295-6

Han, J., Kamber, M., & Pei, J. (2011). *Data Mining: Concepts and Techniques* (3rd ed.). Morgan Kaufmann.

Hashemian, H. M., & Bean, W. C. (2011). State-of-the-art predictive maintenance techniques. *IEEE Transactions on Instrumentation and Measurement*, *60*(10), 3480–3492. https://doi.org/10.1109/TIM.2009.2036347

He, H., & Garcia, E. A. (2009). Learning from Imbalanced Data. *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, *21*(9), 1263–1284.

Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, *20*(8), 832–844.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *Statistics An Introduction to Statistical Learning* (1st ed.). Springer-Verlag New York.

Korst, J., Pronk, V., Barbieri, M., & Consoli, S. (2018). Introduction to Classification Algorithms and their Performance Analysis using Medical Examples. In *Data Science for Healthcare: Methodologies and Applications* (to be publ, pp. 1–35).

Liao, T. W. (2008). Classification of weld flaws with imbalanced class data, *35*, 1041–1052. https://doi.org/10.1016/j.eswa.2007.08.044

Liaw, A., & Wiener, M. (2002). Classification and Regression by randomForest. *R News*, *2*(3), 18–22.

Pete, C., Julian, C., Randy, K., Thomas, K., Thomas, R., Colin, S., & Wirth, R. (2004). Crisp-Dm 1.0. *CRISP-DM Consortium*, 1–76. https://doi.org/10.1109/ICETET.2008.239

Powers, D. M. W. (2011). Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation David. *Journal of Machine Learning Technologies*, *2(1)*(December), 37–63.

Prajapati, A., Bechtel, J., & Ganesan, S. (2012). Condition based maintenance: A survey. *Journal of Quality in Maintenance Engineering*, *18*(4), 384–400. https://doi.org/10.1108/13552511211281552

Provost, F., & Fawcett, T. (2001). Robust Classification for Imprecise Environments. *Machine Learning*, *42*(1), 203–231.

Provost, F., & Fawcett, T. (2013). *Data Science for Business* (1s ed.). O'Reilly Media.

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing and Management*, *45*(4), 427–437. https://doi.org/10.1016/j.ipm.2009.03.002

Susto, G. A., Schirru, A., Pampuri, S., McLoone, S., & Beghi, A. (2015). Machine learning for predictive maintenance: A multiple classifier approach. *IEEE Transactions on Industrial Informatics*, *11*(3), 812–820. https://doi.org/10.1109/TII.2014.2349359

Wirth, R., & Hipp, J. (2000). CRISP-DM: Towards a standard process model for data mining. *Proceedings of the Fourth International Conference on the Practical Application of Knowledge Discovery and Data Mining*, (24959), 29–39. https://doi.org/10.1.1.198.5133

# Appendix A – Construction of a PM Schedule

*This appendix may be undisclosed due to confidentiality reasons.*

# Appendix B – The Monitor Performance Test

*This appendix may be undisclosed due to confidentiality reasons.*

# Appendix C – Data Sources

*This appendix may be undisclosed due to confidentiality reasons.*

# Appendix D – Raw Data Cleaning Report

*This appendix may be undisclosed due to confidentiality reasons.*