

MASTER

Data-driven tool for the implementation of condition-based maintenance

Mrait, M.

Award date:
2019

[Link to publication](#)

Disclaimer

This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain

Data-driven tool for the implementation of condition-based maintenance

Master Thesis

by Marwan Mrait BSc

In partial fulfillment of the requirements for the degree of Master of
Science, in Manufacturing Systems Engineering

Supervisors:

dr.ir. H. (Rik) Eshuis	TU/e, IS
dr. W.L. van Jaarsveld	TU/e, OPAC
Wouter van Dis	Fokker Services B.V.
ir. Ahmet Taspinar	Fokker Services B.V.

Monday 11th February, 2019

Abstract

The main research objective of this thesis is to explore the possibilities that prescriptive data analysis can provide, to support the maintenance decision making process at a Dutch aerospace company. A thorough description of the current maintenance policy and processes, and the potential savings of implementing a condition-based maintenance (CBM) policy are presented. This research stipulates the development, evaluation and deployment of a modeling tool, that can predict the remaining useful life of an elektro-mechanical component.

Executive Summary

Background information

Fokker Services is a global aerospace specialist, and is one of the four business units of Fokker Technologies. Fokker Services is the unit that focuses on aviation component maintenance, repair and overhaul (MRO) services. Fokker Services is continuously aiming to adapt and optimize its maintenance policies, either as way to connect to its customers by providing higher maintenance quality or as a way to lower operations cost. As a part of its long term strategy to uphold their level of reliability and availability, Fokker Services has the intention to invest in the implementation of a condition based maintenance policy for company wide use. Fokker has reason to believe that the implementation of condition based maintenance (CBM) will help them improve their maintenance practices. The Advanced Analytics department has been tasked with the development of this policy by launching a series of projects that aim to adopt emerging techniques in the field of machine learning and business intelligence.

CBM is a preventive maintenance policy, which bases replacement actions not on time or usage, but on the health of the system that is monitored. By monitoring measurable parameters in the system, e.g temperature or vibration, decision on when to perform maintenance actions can be made. As Geitner and Bloch (2012) state, 99% of all machine failures are preceded by detectable indicators.

Research questions

Based on the problem context and discussion with the advanced analytics department and the maintenance engineers from the Schiphol shop, the following main research question is formulated:

“How can we accurately predict the condition of a specific elektro-mechanical component based on available data?”

This research questions captures two aspects; a data analysis of failure and component specific data, and a linkage to condition based preventive maintenance.

Method of gathering data and analysis

The objective is to choose a prognostics tool based on the comparison of several classifiers. This tool is created on a computational modeling environment called `Python`. This language is used because it is the standard programming language of the advanced analytics department at Fokker Services. To get to an answer of the research question I followed the widely used cross-industry standard process for data mining; CRISP-DM (Shearer C., 2000). This is a process used to tackle problems in the field of data analytics. CRISP-DM breaks the process of data mining into six major phases. During the first phases a great amount of time is spent on making the data suitable for analysis. Figure 5.1 shows the schematic overview of the data collection procedure.

The analysis is performed by writing and comparing three commonly used machine learning techniques in `Python`: XGBoost, random forest, and logistic regression. The choice for these techniques is based on a ranking of different supervised learning algorithms on criteria as: prediction accuracy, interpretability, performance with small data sets, and other criteria.

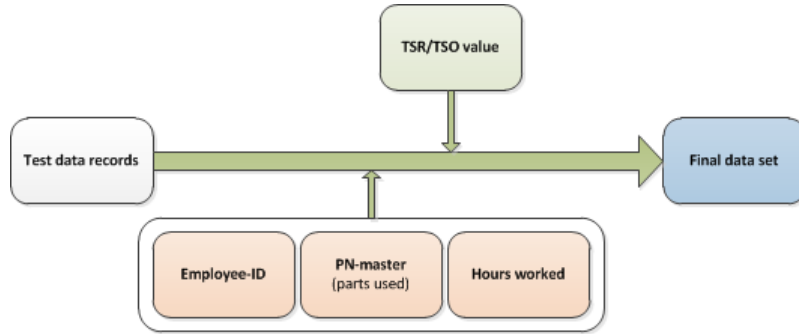


Figure 1: A schematic overview of the data gathering procedure.

Overview of main findings

The models have an average accuracy (0.56 – 0.73). XGBoost and Random Forest score significantly higher than the Logistic Regression estimator. This is because the aforementioned classifiers are ensemble tree learners that select important features automatically. What is striking is that the performance for the class; $TSR = 4000$ is lower than the other two classes. The models are less able to make the classification at the separation line of 4000 flight hours, see Table 1.

Comparing results from the three classifiers I draw the following conclusions:

- The performance of the XGBoost model is slightly better than those obtained for random forest and logistic regression, both exhibiting quite similar results.
- XGBoost model is able to reach good precision (p) values for both "low TSR" ($p = 0.83$) and "high TSR" ($p = 0.75$) values.
- On the other hand, the recall (r) for the XGBoost model is very low ($r = 0.56$ and $f1 = 0.63$); this means that the probability of false alarm for this classifier is quite high (in $1 - 0.56 = 44\%$ of the cases, the classifier raises a false alarm).

Classifier		Evaluation metric		
		Precision	Recall	F1-score
Random Forest	TSR = 2000	0.69	0.63	0.65
	TSR = 4000	0.50	0.58	0.53
	TSR = 6000	0.63	0.65	0.66
XGBoost	TSR = 2000	0.83	0.74	0.75
	TSR = 4000	0.63	0.56	0.66
	TSR = 6000	0.75	0.77	0.77
Logistic Regression	TSR = 2000	0.62	0.61	0.60
	TSR = 4000	0.56	0.56	0.56
	TSR = 6000	0.65	0.63	0.61

Table 1: Classifier performance after parameter optimization

At the end of the evaluation phase, it was concluded that the tool is not ready for effective deployment in the repair shop. The tool and the results of the Evaluation phase will be used as input for a follow-up study on CBM application at Fokker. The feature importance ranking however, does provide unique insights on the underlying failure mechanism and will be used to improve the test procedures.

Suggestions for future research

For future research there are several areas where additional research can lead to insights to improve results in terms of predictive maintenance, prediction accuracy and cost savings. These areas are as follows.

The first step that is necessary to implement CBM is to decide on which data to collect, i.e. what should be considered as input data in the CBM program. In this step it is important to get a thorough understanding of the system in question and decide whether the current collection procedure specified in the CMM manual is sufficient for remaining lifetime estimation or if new test readings need to be logged. This can be seen in my analysis of the correlation of the features with the component condition, where I could not find any correlations with the TSR. In this case it might be needed to install additional hardware to be able to log other signals that are currently not available. The current situation at hand in the analytics department is the availability of a lot of data for analysis, but a lack of knowledge about how to interpret the data. This could be called “data rich – information poor”.

During my conversations and research at the Schiphol repair shop, it became clear that the use of text and notes is widespread in maintenance documents. These notes often contain important information about the failure mode and the repair tasks that have been taken. Using modern techniques, this information can be converted into useful input data to predicting future failures. Text mining is a well-known technique that can be used to detect patterns and tendencies. Concretely, texts will be automatically structured, dissected, transformed, and subsequently inserted into databases where it can be evaluated and interpreted. In my opinion, investing in text mining can generate a large amount of important data for the application of CBM.

A cost-benefit analysis on the trade-off between false alarms and false negatives should be conducted. Finding the optimum balance between these two classification errors that results in minimum costs can provide future developers of the tool information to set certain parameters of the model in such a way that this balance is reached. In order to make this analysis, the different costs and prices for making the errors along with their respective probabilities should be investigated.

Contents

Contents	vii
1 Introduction	1
1.1 Problem Definition	1
1.2 Report structure	2
2 Literature	4
2.1 Maintenance policies	4
2.1.1 Corrective (breakdown) maintenance	5
2.1.2 Planned maintenance (PM)	6
2.1.3 Condition based maintenance	7
2.2 CBM in the aviation industry	7
2.3 Implementation process of a CBM program	9
2.3.1 Data acquisition	9
2.3.2 Data processing	10
2.3.3 Maintenance decision making	10
2.4 Conclusion	11
3 Research Methodology	12
3.1 Research question	12
3.2 Scope	12
3.2.1 Business relevance	12
3.2.2 Scientific relevance	13
3.3 Project assignment	13
3.4 Project approach: CRISP-DM	14
3.4.1 Business understanding	15
3.4.2 Data understanding	16
3.4.3 Data preparation	17
3.4.4 Modeling	17
3.4.5 Evaluation and Deployment	18
3.4.6 Project outline	19
4 Business Understanding	20
4.1 Fokker Services	20
4.1.1 Advanced Analytics	20
4.1.2 Electro-mechanical component	21
4.2 Business Objectives and Success Criteria	21
4.3 Data Mining Goals	22
4.3.1 Classification	22
4.3.2 Performance measures	23
4.3.3 Initial Assessment Tools and Techniques	24

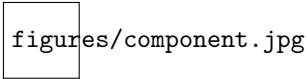
5	Data Understanding	25
5.1	Data Collection	25
5.1.1	Data Sources	25
5.1.2	Collection Procedure	25
5.2	Data Description	26
5.2.1	Summary	27
5.2.2	Initial Insights	27
5.3	Data Quality	29
5.3.1	Accuracy	29
5.3.2	Completeness	29
5.3.3	Timeliness	29
6	Data Preparation	31
6.1	Feature Selection	31
6.2	Data Cleaning	31
6.2.1	Work order duplicates	31
6.2.2	Missing values	32
6.2.3	Encoding of categorical variables	32
6.3	Data Integration	33
7	Modeling	35
7.1	Modeling Assumptions	35
7.2	Model Selection	35
7.3	Test Design	36
7.4	Neural Network	37
7.5	Decision Trees	40
7.5.1	XGBoost	40
7.5.2	Random Forest	43
7.6	Logistic Regression	45
7.7	Model Assessment	46
7.7.1	Revised Parameter Settings	48
8	Results	50
8.1	Main Findings	50
8.2	Feature Importance	52
9	Conclusions and Recommendations	55
10	References	57
	Appendix	59
A	Fokker organigram	60
B	Slip test	61
C	Python Models	62
C.1	Neural Network	62
C.2	XGBoost	63
C.3	Random Forest	64
C.4	Logistic Regression	65
D	Scikit-learn flowchart: model selection guide	66
E	XGBoost Classification Trees	67

Chapter 1

Introduction

The research of this thesis is conducted at both the Fokker's component Maintenance, Overhaul and Repair department (CMRO) located at Schiphol and the Fokker Services headquarters at Hoofddorp. CMRO functions as a repair shop and is responsible for ensuring a certain service level in terms of availability and reliability of individual components for all kinds of aircrafts; Boeing, Airbus, Bombardier, but also of its own Fokker series. CMRO consists of three distinct repair shops, based on the category of the components; avionics (electronic systems used on aircrafts), hydraulics and power generation. All the financial, sales and marketing functions are performed at the Hoofddorp headquarters. The advanced analytics department, that focuses on using data intelligence and insights, is also located in Hoofddorp.

Fokker Technologies (FT) has four business units; Fokker Aero structures, Fokker ELMO, Fokker Landing Gear and Fokker Services. Fokker Services focuses on providing availability services with an emphasis on reliability and maintenance operations of its components. These operations include repairing, testing and overhauling (MRO) services. Fokker has the ambition of being the world's most innovative aerospace service provider by increasing their technical dispatch reliability while reducing direct operating costs. To attain this goal, Fokker Services must ensure its maintenance policy is up-to-date with the ever-increasing expectations of their customers and the seemingly growing benefit and relevance of data science and big data.



figures/component.jpg

Figure 1.1: Integrated Drive Generator (component)

1.1 Problem Definition

Fokker Services distinguishes two customer categories; ABCACUS and commercial customers. ABACUS is a lease service program where Fokker is responsible for the maintenance operations and decisions of the components. This includes the decisions on when a component needs to be brought in for overhaul in the shop. In return, Fokker invoices the customer on a power-by-the-hour basis and promises a certain level of reliability and availability. This promise does not only rely on performing maintenance actions on the components, but also on the readily available pool of spare parts that can be used when needed. The benefit for ABACUS customers is the ability to limit their inventory on location and have access to a wide selection of components in the exchange pool. The commercial customers bring in their components for repair or overhaul themselves and expect a certain level of reliability on the repaired parts.

The preventive maintenance procedures of Fokker have a big impact on the total cost of ownership. A sub-optimal or bad maintenance strategy will either lead to conducting preventive

maintenance too early or too late. In both cases, Fokker pays for the extra maintenance costs. Therefore, there is an incentive to continuously improve maintenance policies by gathering and analyzing data on the component to avoid unnecessary and costly maintenance actions. This is where condition-based maintenance (CBM) and data analytics come into play.

Problem statement

The current maintenance policy for the component is a usage-based policy. A limit to the number of flying hours (soft time) is set, and the components are preventively sent to the repair shop in Schiphol when it reaches this threshold. The soft time is chosen based on historical failure data of the component and may be adjusted to fit the operational schedule. A difference with hard time, where the component requires a specific action (overhaul, repair, test, etc.) at a specific interval, is that soft time is a recommendation by the service provider. In this case Fokker Services. The component may or may not be sent to the shop when it reaches the threshold usage. However, as shown in the master thesis of (Claessens, 2017), a large portion of components is sent to the shop for inspection after an unscheduled down (failure), see Figure 1.2. Claessens calls this phenomenon ‘early failures’ and ascribes this to the poor maintenance quality and the burn-in failure phenomenon. This phenomenon has many possible causes. But the result is the same, corrosion and wear of the mechanical parts in the gearbox which will eventually lead to failure of the component. In this research, we are interested in finding the drivers for these failure and possible predictors that can signal an early failure. One can state that the main cause of this problem is that the soft time rule, as it is applied in the current maintenance strategy, does not seem to work for every component. A model that can more accurately determine whether an individual component will fly a certain number of hours could enable Fokker to make data-driven decisions on when to send a component to the repair shop and distinguish between the component’s in the pool based on their ‘observed condition’. This model can improve the current situation by decreasing unnecessary costs that are made when using a soft overhaul time instead of a predicted lifetime.

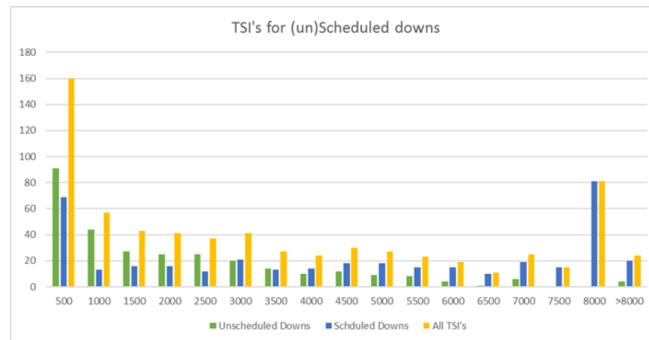


Figure 1.2: Distribution of the Time Since Inspection’s (TSI) for the component

1.2 Report structure

The remainder of this report is structured as follows; Chapter 2 covers the literature on different maintenance policies and the position of CBM in the aviation industry. It also covers the various steps of the theoretical CBM program implementation framework. In Chapter 3 the research methodology is presented. The CRISP-DM framework that was used as a baseline for this project is discussed extensively in Chapter 3. The project outline and the various deliverables of each step are also reported in Chapter 3. Business Understanding, which is the first phase of the CRISP-DM

is covered in Chapter 4. Followed by Data Understanding and Data Preparation in Chapters 5 and 6 respectively. The various modeling techniques that were used in this project are explained and discussed in Chapter 7. This Chapter also includes the test design and the final parameter settings for the various models. In Chapter 8 the results of the prediction models are presented, along with an overview of the important features and a comparison of the chosen models. Finally, in Chapter 9 the conclusion of this project and the recommendations for future research are presented.

Chapter 2

Literature

In this chapter I will discuss the different maintenance policies by citing scientific publications. An important note: the literature on maintenance optimization is vast, both theoretical research as practical case studies are a hot topic at the moment. Therefore I will try to limit our discussion on the research conducted on the aviation industry. In section 2.1 I will start with a brief summary on the existing maintenance policies and provide examples from literature. This is followed by a substantiated reasoning on the merits of CBM and its applicability in different industries. In section 2.2 I will extend the discussion on its application in the aviation industry. This entails recent practices, terminology used in the aviation industry, and the feasibility framework of CBM. In section 2.3 I discuss the implementation process of a CBM program, followed by a conclusion on the findings in section 2.4.

2.1 Maintenance policies

Maintenance operations consist largely of replacing parts of equipment. Maintenance strategies determine when parts or equipment need to be replaced or maintained. In this section I investigate the existing literature on maintenance policies and its development over time. The focus is on data driven maintenance for soft failures. In contrast to hard failures, soft failures lead to systems that will not stop working but will continue to operate at lower performance (Taghipour and Banjevic). This kind of failure is very common in the aviation industry because of the use of backup systems and spare systems. Therefore, it is usually discovered during inspections. Soft failures can however, lead to hard failures if left unaddressed. That is why preventive maintenance is performed; to restore the condition of a system to a satisfactory level and thereby preventing hard failures in the system. Several maintenance policies were introduced and have been implemented in the industry. Corrective or breakdown maintenance is the earliest and simplest maintenance policy. According to this policy, the user of a system only acts after the system or component breaks down (Jardine., Lin, and Banjevic). With the rise of complexity of systems due to the technological advancements, reactive maintenance policies became very costly. Machine availability is of increasingly high priority, therefore long downtimes had to be dealt with. Moreover, some systems that are not allowed to breakdown because of hazardous and deadly situations must have a preventive maintenance plan to ensure 100 percent system availability. And this is especially the case if we look at the high quality and safety standards and regulations set by the aviation industry.

That is why a new branch of maintenance policies was introduced, called preventive maintenance (PM). There are two famous examples from this group of policies that are well documented in maintenance literature; usage, or age-based maintenance and more recently condition-based maintenance (CBM). Figure 2.1 gives an overview of the different maintenance policies and sub-policies. I will briefly discuss each branch separately. Traditional preventive maintenance policies have become unsuitable and inefficient for modern cases (Lin et al., 2012).

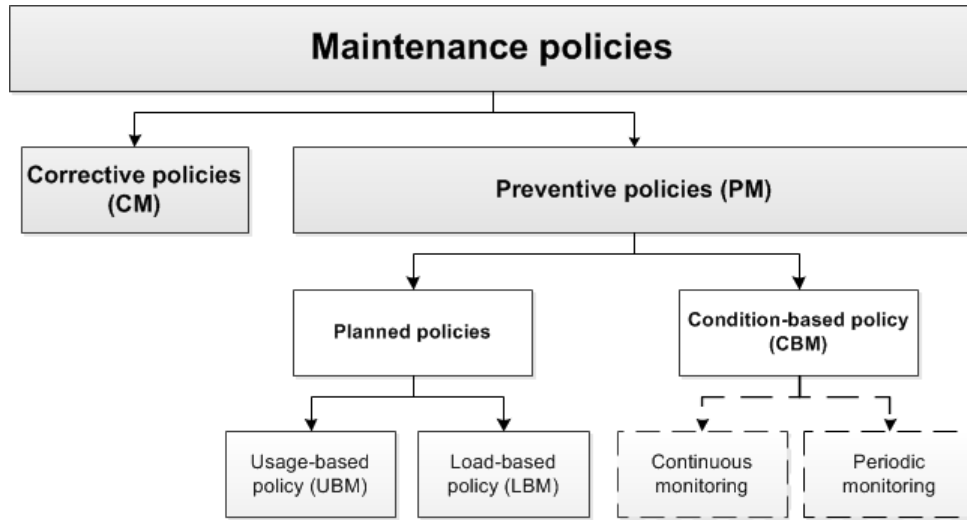


Figure 2.1: Overview of maintenance policies

2.1.1 Corrective (breakdown) maintenance

Corrective Maintenance is equivalent to Breakdown Maintenance. Other common terms for this maintenance policy include Reactive Maintenance and Run-to-Failure Maintenance. Corrective (breakdown) maintenance consists of repairing equipment or assets once it has already failed. A failure of a system can be defined as the inability of a system to meet a specified performance standard. This is called a functional failure (Corrosionpedia, 2018). Functional failures can be apparent or hidden. Hidden failures are functional failures that share two very important characteristics. Firstly, they can't be seen by the operators during normal operation of the system. Secondly, they are usually in items that protect people from severe injury or death, or protect equipment from severe damage. The combination of those two characteristics, and the fact that the corrective maintenance policy stipulates maintenance action after a certain failure is detected, means that there must be some sort of hidden failure-finding task in place (sensors, monitors etc). This is not the case with apparent failures where the consequences of the failure is observed immediately and corrective maintenance action can take place at each observation.

It is the simplest form of maintenance and requires the least planning. However, the resources that it saves in planning and day-to-day operations are often considerably strained due to unpredictable, frequent, and severe breakdowns. Breakdown corrective maintenance is an attractive option for components that do not wear, such as electronics. These type of components have a decreasing- or constant failure rate (DFR or CFR). The failure based policy is optimal for these type of components because performing preventive maintenance does not actually improve the reliability of a component. However, even if a system or component has an increasing failure rate (IFR), the failure based policy can be attractive if the failure rate does not grow without bound and the difference between the corrective and preventive maintenance costs is sufficiently small. Corrective maintenance can be a beneficial strategy when the system or component that is being maintained is:

- Not critical to daily operations
- Not very expensive to repair
- Not very likely to fail

2.1.2 Planned maintenance (PM)

In complex systems, where component damage is too costly or machine availability is vital, corrective breakdown policy is not the optimal policy. Especially in the aviation industry, where functional failures can be catastrophic. Proactive maintenance is the historical successor of corrective maintenance, and also in terms of advantages and drawbacks, PM is the next step following corrective maintenance (Davies, 1990). Usage of PM gives maintenance engineers the possibility to implement maintenance planning and control. PM will ultimately result in reduced downtime, which is the biggest drawback of corrective maintenance. The disadvantages of PM are the opposite of the corrective maintenance advantages; the operating costs for a PM policy are relatively high since a lot of maintenance is unnecessary (called over-maintenance). Next to that, component life is not fully utilized in PM, whilst for corrective maintenance it is fully exploited. Finally, and not unimportant, PM does not remove all random breakdowns as they can happen before a planned preventive maintenance action is executed (Martin, 1994).

Maintenance operations are subject to considerable uncertainty. For a particular maintenance action there can be uncertainty both with respect to its timing (When will we perform a maintenance action?) and content (What maintenance will we do during a maintenance action?). The different maintenance strategies including the corrective (breakdown) policy, are organized according to these two uncertainty dimensions in Table 4.1

	Timing known	Timing unknown
Content known	Usage- or age based maintenance	Condition based maintenance (<i>Condition monitoring</i>)
Content unknown	Condition based maintenance (<i>Periodic inspections</i>)	Breakdown corrective maintenance

Table 2.1: Maintenance strategies organized by timing and content

The breakdown corrective maintenance policy has been discussed. In this policy, both the timing and the content of the maintenance actions are unknown variables. Under usage based maintenance, the total usage of a part is measured and maintenance is conducted when a certain threshold level had been reached. The usage of parts can be measured in many ways depending on the nature of the equipment or part at hand. The amount of time (hours, days or weeks) in the field is perhaps the most common way to measure usage. That is why it is also referred to as age based maintenance. Other examples are the mileage of vehicles, and the number of landings for the landing gear of an aircraft.

Martin (1994) proposed a more in-depth method of condition monitoring of machines. He used two relationships in his model, see Figure 2.2. The first relationship is between the usage of the equipment and the internal loads that cause degradation. The second relationship is between the (accumulated) loads and the machine condition and ultimately, remaining life. Since the usage of equipment is usually scheduled, it is possible to plan when this threshold is reached. This prediction is used to plan the moment of maintenance actions. The drawback of using these relationships are the uncertainties that are implied. The technicalities between usage-to-load have to be understood, similarly for the model estimating load-to-life. This means a thorough understanding of the system and the underlying failure mechanism in relation to load have to be obtained. This comes down to the creation of accurate physics models, which is a time consuming and difficult task. A very common, and widely discussed practice when implementing predictive maintenance policies, is the possibility to interchange or repair several parts simultaneously (block replacement/overhaul). This is beneficial when there is a large set-up cost associated with maintenance (Tinga, 2010). Usage based policies can be planned ahead of time, whereas breakdown corrective maintenance cannot be planned at all. As a consequence of this, the resources needed for usage based maintenance can be utilized more fully than resources needed for corrective maintenance.

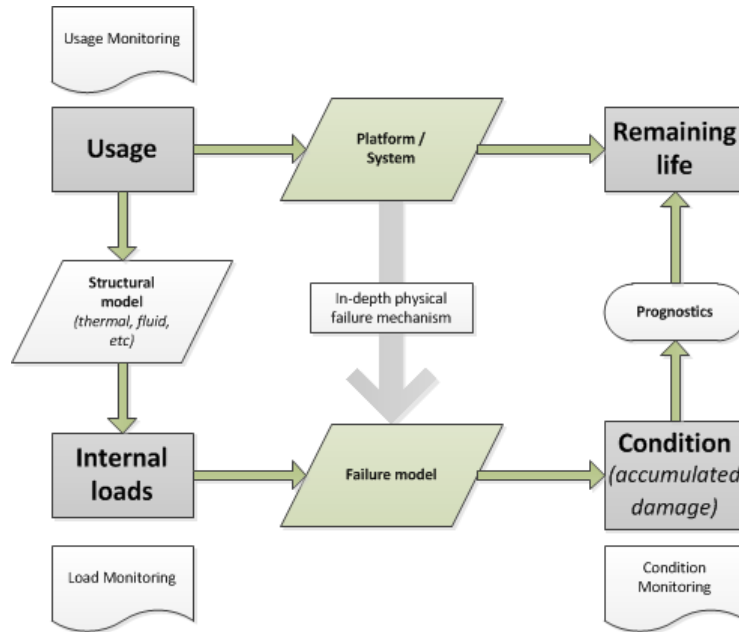


Figure 2.2: Representation of the relation between usage, loads, condition and remaining life (Tinga, 2010).

2.1.3 Condition based maintenance

CBM is a hybrid policy not based on failures or times as its predecessors, but purely based on the health of the system being monitored. It differs from other policies in the PM category in the sense that its predictions are based on the condition or ‘health’ of the system, and not time. Geitner and Bloch (2012) state that almost all mechanical failures in machinery are preceded by a detectable indicator. There is a trade-off in applying CBM policies; if the threshold to perform maintenance action is too high (c.q. the control limit is close to the failure threshold), then there is a chance that the failure is not detected and this increases the chance of an expensive corrective maintenance action. On the other hand, if the control limit is too low, the policy will trigger maintenance action much earlier than necessary. Which is also expensive. Finding an optimal control limit is crucial to minimize the maintenance cost and benefit from the advantages of CBM. Zhu, (2015) has an extensive review on the statistical methods of determining the optimal control limit.

CBM also has some drawbacks, which is one of the reasons that CBM is not adopted globally by companies. The set-up of a CBM program is technologically challenging and very costly. Complex system monitoring systems have to be installed, that are operated by trained maintenance engineers. These high initial investments are the main reason companies still choose other maintenance policies over CBM (Martin, 1994).

2.2 CBM in the aviation industry

Commercial air transport has long been seen as the most safest mode of transportation. Research by Savage (2013) shows a fatality rate of 0,07 per a billion passenger miles. This is very low if we compare it with the most popular mode of transportation, which has a rate of 7,28 fatalities per a billion passengers miles. The research also shows a slow but steady decrease of fatalities in commercial aviation since 1970. It is no surprise then, that safety was and still remains a key factor in the aviation industry. The annual report by the European Commission (2017) shows a

continued growth of costs in the aviation industry, largely driven by fluctuating oil prices. With labour (i.e. salaries, wages, and benefits) and fuel costs being more challenging to control, airlines focus significant attention on maintenance. The expenditure in MRO activities in 2015 was 3.5% higher compared to 2014 (Michaels, 2016).

One of the emerging topics in the aviation industry, is the discussion of how technological advances are expected to impact the MRO industry. Wyman (2016) says that the emergence of Big Data and the global trend of using data science tools, will drive significant changes in maintenance frequencies as well as in maintenance methodologies. This growth is extra driven by the introduction of new generation aircraft. In the next decade the fleet of new generation aircraft will grow by approximately 530% to nearly 19.000 globally, according to the annual ICF International report of 2017. Aircraft with new technology challenge traditional MRO sourcing strategies. According to Michaels (2016), there is a cost saving due to fewer routine air-frame maintenance, compared to the (older) Boeing 767 aircraft. The new generation Boeing aircraft also has a higher asset utilization: an additional 90 days of available flying hours. Figure 2.3 shows the 12-year heavy maintenance schedule of both generations.

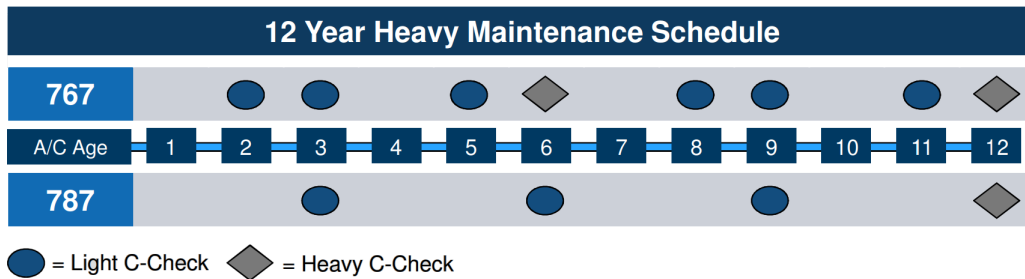


Figure 2.3: 12-Year heavy maintenance schedule of the Boeing 767 and 787 aircraft. Source: Michaels (2016)

A literature study by Van den Bergh et al. (2013) stated that the majority of the operations research in the aviation industry is focused on flight and crew scheduling and not so much on maintenance optimization. From the 102 papers reviewed, only 8 papers mention CBM in some case. However, contributions to the field of aircraft maintenance and component usage optimization are growing. Most literature on the application of CBM can be divided into two categories. One category focuses mainly on the quality of CBM policy implementation in the aviation context, and does not actually add to the research in this area to prove CBM usefulness. These articles propose to enhance the quality of CBM programs and try to give guidelines for better CBM execution. An analysis on the application of prognostics and health management, a term similar to CBM, was proposed by Wen and Liu (2011). Another article by Zhao et al. (2012) proposed a so called binary decision diagram (BDD) to enhance maintenance decision making of aircraft engine components. By applying the BDD, sensor data from the aircraft monitoring network is sent to the control center located on the earth's surface. Other articles in this category discuss methods of improving the reliability of the monitoring network and promote satisfying CBM implementation within the aviation industry.

The other category is more about the process on implementing a CBM program for aircraft components. A prominent contributor on this subject is Carl Byington, with four articles on CBM implementation on aircraft components. In one article, Byington and Stoelting (2004) developed a prognostic-based model to predict the degradation on a flight connector actuator. In the same year another article was published in which the remaining life was predicted using data-driven neural networks (Byington et al. 2004). Both papers are a good example of how data-driven prediction models are used on aircraft components. Sun et al. (2012) presented a state space model (SSM) for prognostics on a gas turbine subject to degradation. A time-to-failure distribution was found by fitting a curve using Bayesian state estimation.

It is notable, that prognostic-based models that predict the time to certain failure states and the degradation process, are often disregarded in conventional machines and systems. These models that need large investments and have high technological requirements are used more often in the aviation industry. Philips et al. (2010) state that this is due to the fact that knowledge on the fault propagation and system failure mechanics is required by aviation authorities. This guides the choice to implement the more sophisticated model-based approach. This is not always the case. Knowledge on the underlying failure propagation is lacking for certain components, as will be shown with the component used by Fokker. Other studies on CBM implementation in the aviation industry are: Jianhui Luo et al. (2008), Philips and Diston (2011) and Roemer et al. (2001).

2.3 Implementation process of a CBM program

Various authors in the field of CBM propose to use a systematic approach by using a framework that entails all necessary steps for a CBM program. The most prominent framework that is cited and used by researchers is the framework of Jardine et al. (2006). It consists of three main steps; data acquisition, data processing and maintenance decision making. There are some slight deviations found in other works. Peng et al. (2010) for example, states that the prognostics during the maintenance decision making process can only be done after diagnostics. Jardine et al. (2006) acknowledge that both diagnostics and prognostics are part of the last steps, but they are not necessarily dependent of each other. A systematic overview of the different (sub)steps- and components of the implementation process of a CBM program are shown in Figure 2.4

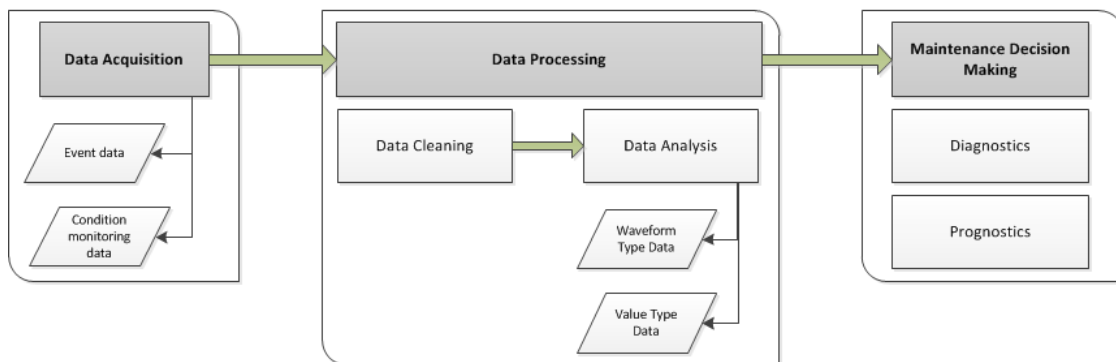


Figure 2.4: Standard CBM program implementation framework

2.3.1 Data acquisition

The first step of each CBM program is collecting relevant data that can be used to assess system health. Data collection can be done by numerous means and the data can originate from various sources. The technological developments in the last decade in the field of sensors, advanced signal processing, and early detection techniques have made condition monitoring and CBM more and more affordable (Engel et al., 2000). The most common sensors used to collect information on the condition of systems are; thermal sensors, humidity sensors, electrical voltage and current sensors, mechanical vibration sensors, chemical sensors and optical sensors (Pecht, 2000). These sensors provide different types of data that have to be analyzed and processed accordingly.

The data collected for a CBM program can be roughly divided into two types; event data, and condition monitoring data. Event data is all the data that contains information related to a specific maintenance action, or event. These events can be repairs, overhauls or inspections.

The data collection process has also been automated by computerized maintenance management systems (CMMS) or already existing enterprise resource planning systems (ERP). These systems have been developed to automate data storage and handling, and are less time consuming and less prone to errors compared to manually entering the data. This is especially beneficial when keeping track of data sets with large amounts of data. One important statement made by Jardine et al. (2006) is that event data is equally important as condition monitoring data. It is a widely held, incorrect belief, that condition monitoring data is superior and makes event data redundant for CBM processes. This is untrue, because event data does provide feedback to system designers on how to redesign the system and improve the condition indicators. Wang and Chirster (2000) state that the combination of event- and condition monitoring data is what makes CBM more worthwhile.

2.3.2 Data processing

There are a number of different data processing techniques discussed in the literature. All of which are associated with different data categories (or types). I have already mentioned the division between event data and condition monitoring data. Jardine et al. (2006) does not differentiate in the treatment of both data types, but Heng et al. (2009) does, and he also states the advantages and disadvantages of analyzing both types separately. Analyzing event data is the traditional way of obtaining reliability models. A time-to-failure distribution is often generated using a parametric estimation. This approach is widely used to this day, because its simple, does not require the need of collecting condition monitoring data, and can be used for longer range predictions (Heng et al., 2009).

In the context of data processing, condition monitoring data can be further divided into three categories:

- Value type data: This is data from a single value within a specific time epoch. Examples are temperature and humidity. In CBM, value type data can be extracted directly as raw data from sensors, but it can also be acquired from waveform or image data. A widely used technique to process value type data with a large number of variables is principal component analysis (PCA). The main function of this technique is to retain the most important features from multivariate data sets in order to reduce the dimensions and make it more suitable for modeling purposes (Peng et al., 2010).
- Waveform type data: This is data collected during a specific time epoch, also called 'time series'. Vibration of acoustic data is an example of waveform type data. Analysis of waveform type data in the time-domain covers the signal in the original form. Unlike the analysis of waveform type data in the frequency-domain which is done through spectrum analysis. Moving from the time-domain to the frequency domain is achieved by applying fast Fourier Transform (FFT). Details of this application can be found in the recently added article by Yang et al. (2016), who applies FFT in order to use condition monitoring data from engine turbines. Trending techniques like ARMA (a combination of auto-regression and a moving average) are also used to analyze waveform type data (Pham and Yang, 2010). Since trending techniques are used to estimate future values, they can be very useful for prognostics in a CBM program.

2.3.3 Maintenance decision making

In order to maximize the effectiveness of a CBM program, there has to be an integrated decision making mechanism that makes the appropriate choices based on the diagnostics and prognostics results. The most researched prognostic is the estimation of remaining useful life (RUL) (Jardine et al., 2006). But the usage RUL alone is not sufficient for decision making, because it does not incorporate other criteria like risks and costs. RUL estimation can be divided into three main categories: experience-based prognostics, evolutionary or trending models, and model-based prognostics. Experience based prognostics often use a statistical approach to fit a time-to-failure

distribution. Trending models are more data-driven and use predictive models, like neural networks (ANN), or fuzzy logic. Finally model-based prognostics is the most complex approach, because it uses physical models that accurately describe the underlying failure mechanism of the component. These decision making approaches are increasing in accuracy and complexity, but also in costs and technical difficulty. A model-based approach is also very component specific and can therefore not be used as a company wide model. Research in prognostics for different failure mechanisms has brought up models for fatigue, creep, corrosion and wear damage (Tinga, 2010). However, these models are created in controlled environments, and do not include external loads and stresses that may happen in real-life situations.

2.4 Conclusion

In this section I discussed the various maintenance policies, the role of CBM in the aviation industry, and its implementation process. Data processing and analysis of the component test data is covered in Chapter 6. For this end I used processing techniques for value type data, because no waveform data are present in the data set. Also, in Chapter 7 I make use of the model-based prognostics described in 2.3.3. These prognostics give a more interpretable view on the underlying physical failure mechanism of the component, which was one of the requirements of this project.

Chapter 3

Research Methodology

3.1 Research question

Based on the problem context and discussion with the advanced analytics department and the maintenance engineers from the Schiphol shop, the following main research question is formulated:

“How can we accurately predict the condition of a specific component based on available data?”

This research questions captures two aspects; a data analysis of failure and component specific data, and a linkage to condition based preventive maintenance.

3.2 Scope

3.2.1 Business relevance

The findings of this research will be used to the benefit of Fokker Services in a broad sense. The current maintenance policy is based on a rule-based preventive maintenance strategy. Multiple studies have shown that data-driven maintenance policies have a better performance in terms of; costs, quality (Lawrence, Anuj, and Gerald, 1995), (Rosmaini and Shahrul, 2012), and inventory management (Srinivasan and Lee, 1996). Competitors in the aviation industry are investing more in big data analytics (McAfee and Brynjolfsson, 2012). This justifies exploring the utility of test data, that has been gathered at Fokker Services since 2010. With the findings of this study, Fokker Services can decide to switch over to a dynamic data-driven maintenance policy, that considers a separate analysis of individual components. Given a certain level of prediction accuracy, sales personnel from Fokker Services can be provided with indicators that can measure the degradation of a component. The output of this study can be used to create a database that assigns each component to a record of historical data, and a prediction of the remaining lifetime. With this extra information, sales agents can allocate certain component’s to certain customers based on the customer requirements and preferences. The result of this research can also be rolled out to other Fokker products with similar testing procedures.

The findings also provide significant benefits to the managers and planners of the Fokker Schiphol repair shop. The final model provides a list of features that are important or significant in measuring the condition or degradation state of a component. These features correspond to specific tests that are conducted consecutively during a test run. Given the importance of these tests, management can decide to pay more attention on the execution of these tests. Fokker is trying to slowly move towards condition-based maintenance with real-time gathering of (sensor)data during test runs. To implement this strategy, there must be scientific evidence in the form of an accurate model that shows the applicability (in terms of reliability, validity and other measures)

of predictive modeling on maintenance policies. The findings of this study may help uncover these important predictors.

3.2.2 Scientific relevance

The application and implementation of condition-based maintenance is a widely discussed topic in scientific literature. Noman et al. (2017) presented an overview of the most influential researches conducted in this field, using bibliometric indicators. These indicators measure the size and impact of research papers and other publications based on the number of citations received. One of the conclusions from his survey was that most papers on condition-based maintenance tend to focus on data-driven approaches. This data can be historically observed failure data, so called event data. This is a widely used source of data in reliability research and it is extensively analyzed for systems with vibrating components by Jardine et al. (2006). However, some systems or components are not able to provide this kind of data as they are not allowed to run until failure for example. This is the case for critical components, the failure of which could have a catastrophic effect upon the system. In the avionics industry, the critical parts are defined as;

“A part identified as critical by the design approval holder. Typically, such components include parts of which a replacement time, inspection interval, or related procedure is specified in the Airworthiness Limitations sections.”-(EASA, 2011)

For these type of parts failure data maybe scarce. Therefore, another data source is widely used; condition-maintenance data or CM data. This type of data is defined as any type of information that may have a connection with the degradation or the estimation of the remaining useful lifetime (RUL). Such as monitored sensor data, performance data and degradation signals (Wang, Chu, and Mao, 2008). Examples of CM data are; vibration data, temperature, oil analysis data, pressure, humidity, moisture, speed, load, etc. The data can be both objective and subjective, based on the nature and the collection method of the data.

Over the last few decades, there have been a number of review papers on data driven approaches in CBM, see for example (Cho and Parlar, 1991) (Reinertsen, 1996) and (Scarf, 1997). According to the literature study there is an increase of studies to the field of aircraft maintenance (Van den Bergh J. , De Bruecker, Beliën, and Peeters, 2013). This growth evolves mainly around three topics, where the concept of condition-based modeling is addressed in some way. These topics are ‘aircraft maintenance scheduling’, ‘task/job allocation’ and ‘optimization of specific components usage’. However, despite the increase of interest in this field, there is only a fraction of articles that discuss the practice of CBM in the aviation industry. Only 8 of the 102 papers in the literature review of Van den Bergh et al. (2013) mention the CBM policy in some sense. The research of this master thesis will serve as a contribution to the list of CBM practices in the aviation industry.

3.3 Project assignment

Based on the defined outline of the project’s objective and purpose, I can formulate the assignment of the project as follow;

“Create a data driven and generic tool that predicts the remaining useful lifetime of the component as accurately as possible.”

Because Fokker Services has the wish to roll out any predictive models to other products in the future, the tool must be as generic as possible. Meaning that it must be able to handle different data sets and perform the same analysis without much modifications. To answer the main research question in a structured manner, I subdivide it in the following sub-questions. The output of each sub-question leads to a result that is used as intermediate milestones of the project;

1. *What are the objectives of the tool?*
 - What are the business success criteria?
2. *What are the available sources and types of data that are suitable to predict the remaining useful life of a component?*
 - What is the format of the existing data (discrete, event/failure, text, categorical)?
 - Is the available data content-wise suited to perform predictive modeling?
 - What are the required pre-processing steps (imputing missing-values, checking data quality and consistency, data cleaning, etc.)?
3. *Which models are appropriate for the prediction analysis?*
 - Literature study of relevant data analysis models (Logistic regression, Decision tree, Fuzzy Logic, Neural Networks and other machine learning techniques)?
 - Elaboration on the applicability and feasibility of the models for the problem at hand (dependent and independent variables).
4. *Which models are appropriate for the prediction analysis?*
 - Assessment of different models with several performance measures; goodness of fit, accuracy, recall, model validation.
 - Evaluation of the prediction error (magnitude, significance) for different values of the ‘soft time’.
 - Analysis on the feature importance.
 - Measure the uncertainty and robustness of the model output by way of a sensitivity analysis.
 - Exploration of an appropriate set of features by conducting a confirmatory factor analysis.
5. *How can the model outcome be used to improve maintenance decisions?*
 - What is the relation between predicted lifetime and the condition of the component?
 - What are the thresholds, in terms of flying hours, of early failures?
 - Exploration of different new data sources; data mining of operator texts, sensor data, event data etc.

3.4 Project approach: CRISP-DM

The nature of this project and the research questions at hand form a good foundation to use data analysis approaches. The main objective of this project was to design a tool that can predict a target variable that represents the condition of the component. This tool was created on a computational modeling environment called `Python`. This language was used because it is the standard programming language of the advanced analytics department at Fokker Services. Some data analysis tasks however, were performed in `R`. This is a software package and programming language that is better suited for statistical data analysis. `R` is an object-oriented language, like `Python`. Its source code is freely accessible and can be used in a variety of applications, including `Alteryx`. To answer the questions of section 3.3 I followed the widely used cross-industry standard process for data mining; CRISP-DM (Shearer C., 2000). This is a process used to tackle problems in the field of data analytics. CRISP-DM breaks the process of data mining into six major phases, see Figure 3.1. The diagram in Figure 3.1 also shows the possible iteration points of the CRISP-DM framework. In this project I have performed four iterations from the Modeling phase back

to the Data Preparation phase. This was necessary in order to prepare the data set for other modeling techniques. At the end of the evaluation phase, it was concluded that the tool was not ready for effective deployment in the repair shop. The tool and the results of the Evaluation phase will be used as input for a follow-up research on CBM application at Fokker. For the remainder of this section I will discuss the contents of the CRISP-DM phases, and how the output of these phases led to an answer to each of the research questions stated in section 3.3.

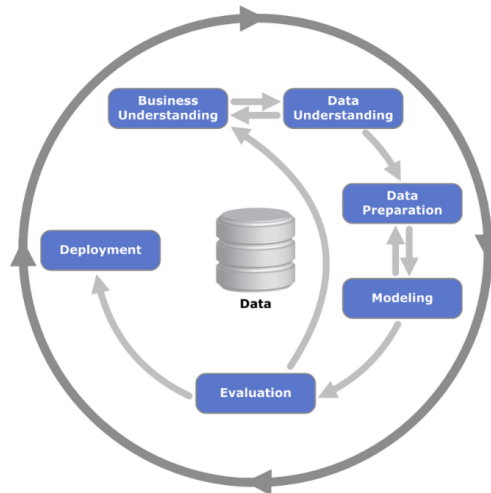


Figure 3.1: Diagram of the CRIPS-DM phases

3.4.1 Business understanding

The first step in the process of building a prediction model for the lifetime of the component was to understand the project objectives and requirements from a business perspective and then convert this into a formal data mining problem statement. This statement was essential and was agreed upon by the supervisors. An agreed upon choice in this project was to make a classification model, rather than a regression model. The reason for this is because it aligns better with the objective of the project; having an accurate indicator of the condition or degradation. The way the remaining lifetime is classified is also a variable; the number of classes, the boundaries of the classes, etc. The business understanding phase consists of four steps.

The first step was to determine the business objectives of this project. This was achieved by conducting interviews with several stakeholders to obtain a thorough understanding of the problem at hand. The business success criteria were also clearly defined during this step. A situation assessment was the next step in the business understanding phase. This step resulted in several deliverables; an assessment of the available resources, model requirements, assumptions and constraints, risk and contingencies, and finally the costs and benefits.

The third step was agreeing upon the data mining goals. Here, the business objectives were translated in tangible and measurable data mining goals. Success criteria were used to evaluate the output of the final tool and compare different solution designs.

The final step of the business understanding phase was the project plan. The project plan consists of an overview of the business objectives and the intermediate deliverables of the project. This project plan was used to monitor the progress during this eight months project. The time line of this project, along with an initial assessment of the different tools and techniques are discussed in the next section. Figure 3.2 displays an overview of the above-mentioned steps.

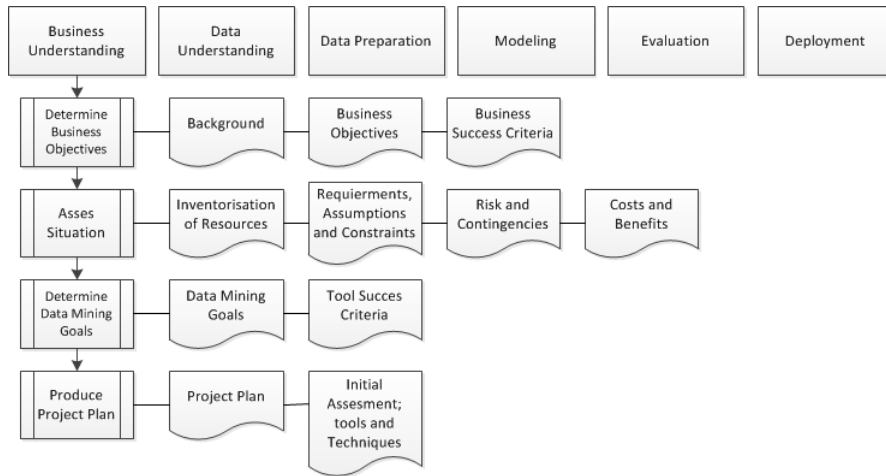


Figure 3.2: Business understanding steps

3.4.2 Data understanding

The process of data collection was not within the scope of this project, because an initial data set was already provided. This data was gathered manually from the Fokker Services database. Historical test data dating back to 2009 was extracted and then exported to separate excel files. The steps and the deliverables of this phase are displayed in Figure 3.3.

The first step in this phase was to describe the available data sources and the procedures of collecting this data. After that we dived into the data and identified data quality problems (missing values, inconsistencies, repetitiveness, typo's, etc.). In this phase I also performed a descriptive analysis of the data to discover first insights and get familiar with the structure of the data. Incompleteness of the data or serious issues regarding the quality of the data were solved in this phase before I proceeded to the next step. The provided excel files contained test data of approximately 300 component work orders. Each work order has a unique serial number and several non-test data variables. In this phase, only the data quality was considered. Issues with usability of the data where dealt with in the next phase.

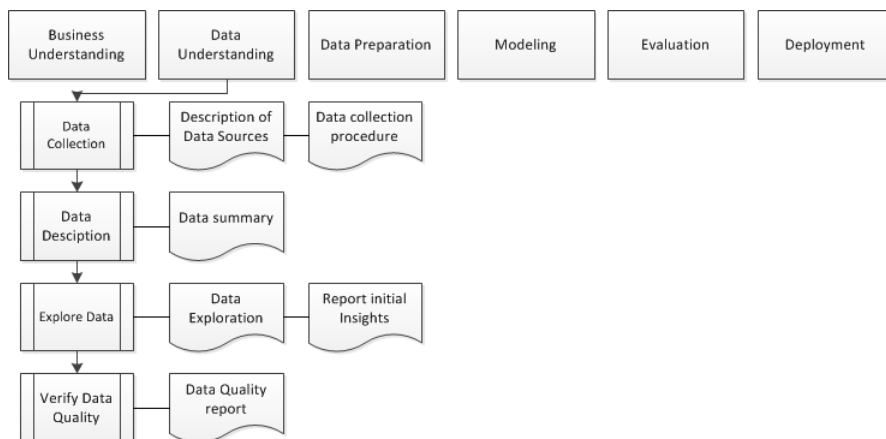


Figure 3.3: Data understanding steps

3.4.3 Data preparation

Preparing the data is an important step in the CRISP-DM methodology. It is also the most time-consuming part of a data analysis project. The following steps describe the tasks during this phase, but it should be noted that these steps were not performed sequentially. These steps do however cover all activities that were needed to construct the final data set, that was in return fed into the modeling tool. The deliverables of this step were to convert the provided raw data into a structured data set in a workable tabular format, see Figure 3.4. This format contained only those variables that are considered useful for the prediction model.

The first step was data selection. The available data was provided with a lot of non-test data. It was important to investigate the usability of this type of data, and why certain data was not suitable to work with. Next, a thorough cleaning process of the data was performed. Missing values were imputed, duplicates were removed, and categorical variables were encoded. It was important to the end-user of the tool to understand why certain cleaning procedures were conducted. Therefore, a detailed documentation of the different steps were made (log file). In the final steps, the data was integrated from different sources into one final data set. After that, the data was also formatted to suit the needs of the modeling tool. The component test data were extracted in separate excel files per work order, the deliverable of in this phase was a merged data set of all (unique) work orders.

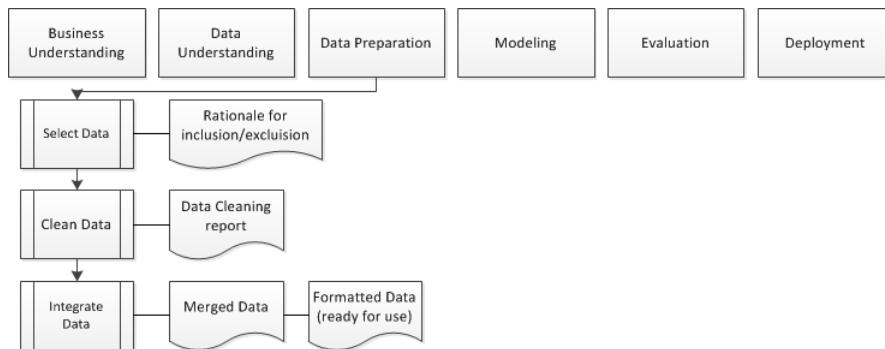


Figure 3.4: Data preparation steps

3.4.4 Modeling

In this phase, classification models were built in `Python` by means of various modeling techniques. Each model had specific requirements on the form of the input data. Therefore, stepping back to the data preparation phase was needed four times. The deliverable of this step were several executable `Python` scripts, see Figure 3.5.

First, an assessment of the different modeling techniques was conducted. The data mining techniques were arranged based on the specific data mining goal: multi-class classification. Although there is a wide variety of machine learning techniques to choose from, I will focus on the following data mining techniques

- Neural Networks
- Gradient Boost trees
- Forest Classification trees

The rationale behind the choice of these specific techniques was based on a comparison of all the available models by a set of modeling criteria, see section 7.2.

In the next step, a test design was generated. The data was split into a train set, and a test set. A description was given on the sampling technique and how the model was built and evaluated. After that, the actual models were built. This was where the mining takes place. The deliverables for this step were the parameter settings, which listed the parameters chosen for the modeling tool, the models, and the model descriptions which describe and interpret the modeling mechanisms.

The final step in the modeling phase was the assessment of the model. The models were assessed from a technical point of view. Did the model do what it was supposed to do? A number of revisions to the parameter settings were performed during this step.

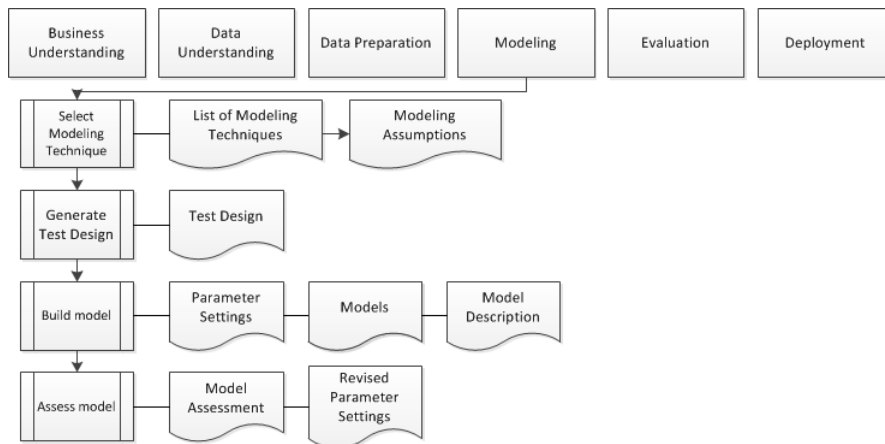


Figure 3.5: Modeling steps

3.4.5 Evaluation and Deployment

At this stage the best fitting model was selected. This was based, not only on model performance (accuracy), but also on the applicability to solve the business problem at hand. Some models appeared to have high quality from a data analytics perspective, but I made sure that the output of the model gave us a valid insight that can be used to take preventive maintenance decisions. Evaluation of the results was done with domain experts; supervisor and maintenance engineers.

The final phase of the CRISP-DM methodology is the model deployment. Creation of the model was generally not the end of a data mining problem. pre-deployment of the tool was on a theoretical new test set. The performance of the tool on this test set, and the costs and benefits of the model predictions ultimately led to a recommendation to Fokker on the implementation of a data driven maintenance policy. A deployment of the tool in the future, could be in the form of a repeatable generic classification tool, that can classify component's in different classes based on the prediction of the remaining useful lifetime. Where each component entering the shop is given a score and a classification label (reaching a certain soft time or not).

The deployment phase also encompassed monitoring and maintaining the tool, to make sure the models worked correctly on new products and data. A final report and presentation were also components of this phase. The report contained a detailed description of all CRISP-DM phases, the rationale of all the decision made, and a visualization of the final data mining results. Finally, a reflection step was made on what went right and what could be improved.

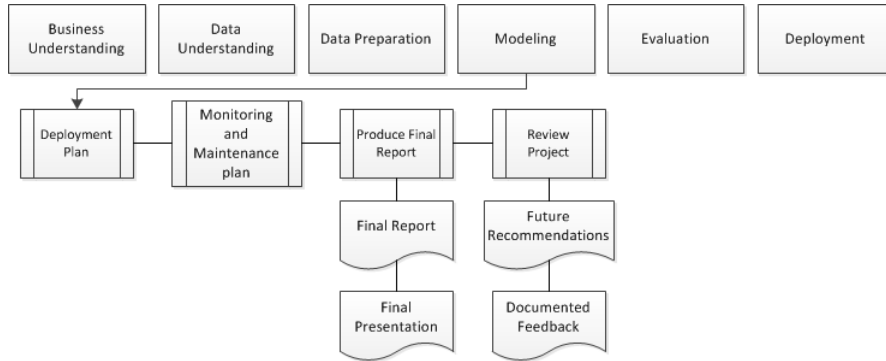


Figure 3.6: Deployment steps

3.4.6 Project outline

The sequence of the phases was not strict and moving back and forth between different phases was sometimes required in the duration of the project. According to literature (Franks, 2007), about 80 percent of the CRISP-DM cycle is data preparation. Especially data cleaning is very time-consuming depending on the quality and the of the data set. Table 3.1 presents an overview of the relation between the research questions stated in section 3.3, and the CRISP-DM phases. Also the required deliverables of each phase is listed.

CRISP-DM Framework		
	Research question	Deliverables
Business understanding	1	Business objectives and success criteria
Data understanding	2a, b	Data summary and initial insights report
Data preparation	2c	Formatted (cleaned) data
Modeling	3	Model descriptions, test design, parameter settings
Evaluation	4	Main findings, feature importance
Deployment	5	Final report

Table 3.1: Relation of CRISP-DM with the research questions

Chapter 4

Business Understanding

This chapter serves as the business understanding step in the CRISP-DM approach. This thesis was conducted at the Hoofddorp site of Fokker Services with a number, in addition to a number of visits to the MRO shop in Schiphol. An introduction of the company is given in section 4.1.1. Next the business success criteria of the project are given in section 4.1.2. Based on these agreed upon criteria, a list of data mining goals is presented along with their respective tool success criteria. These criteria are measurable and are used to judge the feasibility of the final tool. Finally, in section 4.3, the project plan and an initial assessment of the models, tools and techniques are presented.

4.1 Fokker Services

Fokker Technologies (FT) has four business units, one of which is Fokker Services. This unit focuses mainly on the providing availability services with an emphasis on the maintenance, repair and overhaul services (MRO). The other business units of FT are Fokker Aerostructures, Fokker ELMO and Fokker Landing Gear. FT has a total of 19 sites worldwide, of which 5 sites are operated by Fokker Services. At a global level Fokker makes a distinction in three areas. Europa, Middle East and Africa (EMEA) are serviced by the locations in the Netherlands in Hoofddorp, Schiphol and Woensdrecht. The Americas are serviced through the Atlanta site, and Asia/Oceania through the Singapore site. (Fokker Technologies, 2015).

This research is mainly conducted at the component MRO (CMRO) department of the Hoofddorp site. CMRO handles all incoming and scheduled parts for maintenance, overhaul and repair operations. The maintenance actions for the EMEA region are conducted at the Schiphol site. Fokker Services provides CMRO activities for all kind of aircraft types, e.g. Boeing, Airbus, Bombardier and also Fokker's own aircraft components. Zooming-in even further in the MRO department, I see that there are distinct repair shops; avionics (electronic systems used on aircraft), hydraulics and power generation. These are further divided into work centres, based on certain specializations. A (partial) organization chart of the Fokker organization can be found in Appendix A.1

4.1.1 Advanced Analytics

To deal with the growing need of data-driven solutions, Fokker Services launched a new department in the beginning of the year 2017. This department was primarily tasked to forecast future events and behaviors and allowing business to conduct what-if analyses to predict the effects of potential changes in business strategies. This resulted in a series of projects with other business partners and universities, including this thesis. The current manager of the advanced analytics department is responsible for the inception of new projects and facilitating project teams with all the necessary requirements in terms of data and other resources. Working in or with the advanced analytics

department are two engineers that also play a role in this project. The first supervisor is a data scientist within the Advanced Analytics department, and also a company supervisor for this project. The second supervisor is responsible for the development of automated test equipment and in circuit test solutions at the Schiphol shop. Each project starts with an informal conceptual briefing of the problem at hand. And a set of measurable success criteria are agreed upon, see section 4.1.3.

4.1.2 Electro-mechanical component

I already noted that Fokker Services has three distinct repair shops. One of which is the Power Generating (PG) shop, where all power generating components of an aircraft are handled. These are actual generators, as well as auxiliary components assisting in power generation. In the thesis of Claessens (2017) it was concluded that these components were the most suitable candidate for the CBM pilot at Fokker. This was due to a number of reasons:

- **Increasing failure rate.** During interviews with Fokker engineers it became apparent that components in the PG shop experienced more wear and tear due to moving and vibrating parts. Metal friction was also a key contributor to the increasing failure rate. The component also suffered from so-called early failures. This indicates that there is a certain potential for improvement using preventive maintenance policies.
- **Financial feasibility.** Failures in components from other shops (avionics and hydraulics) where mainly occurring in circuit boards and less often in sensors. These kind of failure are less easy to detect compared to the component. Monitoring techniques that measure relevant data on these failures are non existent and very costly to develop. However, due to moving parts and larger size, has a wide range of available condition monitoring techniques. Another point that confirms the selection of this particular component, is the fact that the PG shop maintains much less unique components. This means that the initial investment of a CBM policy (in time and resources) will be much lower than the other shops. It is more cost-effective and technologically less challenging to implement a CBM model on a group of components that are less diverse.
- **Asset criticality** According to expert opinions of Fokker engineers, PG shop components have the highest score in terms of asset criticality. The particular component in question is indispensable for the aircraft's power supply during a flight. Components in the other shops have a lower score because of redundancy and no immediate consequences in case of an in-flight failure. This makes it a more favourable candidate for a CBM program.

Currently, Fokker Services applies an age-based maintenance policy for this electro-mechanical component. It has soft time overhaul time for each component at 8000 flying hours. The component that fails before this threshold are repaired or overhauled correctively. The cost of performing a maintenance action on this component is very high, so with the unnecessary maintenance actions that come along with this policy, there is a great potential for cost saving using a CBM tool. A slight majority (64%) of Fokker's customers are part of the ABACUS program. This is a lease service program, which invoices the customers on a power-by-the-hour basis. Fokker owns and maintains a pool of components and pays for the maintenance actions itself. Therefore an incentive to improve the reliability and maintenance practices, and avoid costly overhauls is present. This is where the CBM tool comes in.

4.2 Business Objectives and Success Criteria

To acquire the business objectives of this project, several people are involved in the MRO activities of Fokker where interviewed. First, a meeting with the department manager was planned where the current situation was explained. The ultimate goal from his point of view, was to create a model that can run by repair shop employees during test runs. This model should use the test

data that comes from the component as input, and the output should be some sort of an indicator on the condition and, if possible, an estimation on the remaining useful life of the component. With the outcome of this indicator, employees should be able to make a more informed decision on what maintenance action to perform. For example if the indicator shows a low chance that the component will reach 4000 flying hours, the repair shop can decide to overhaul the component instead of repairing it. Fokker can also use this indicator to give the component a valuation that is based on the condition of the component. This prevents over- and undervaluation, and with this information they can give clients a better price offering. Because, as stated earlier, unnecessary maintenance action can lead to extra costs that, according to the contract of the ABACUS program, come at the expense of Fokker Services. Another objective of this project is to find out whether it is possible to gain more insights on the underlying failure mechanism of the component. The question is whether certain features in the test data are more relevant than others when predicting the remaining life. From a quality control perspective, this information can then be used to improve the repair and overhaul procedures.

The most important criterion for Fokker Services from the business perspective, is that the model provides a reliable outcome that can be used on the work floor. For example, a model accuracy of 30% on the remaining life is insufficient for practical use. Also, the model should be as generic as possible, in order for a possible integration on other components of the PG shop. The complexity (running time) should therefore be taken into account.

4.3 Data Mining Goals

Now that the business goals are clear, it is time to translate them into a data mining reality. These data mining goals can then be used by Fokker Services to support their maintenance decision making process and reduce their MRO expenditure. These goals are summarized in the following bullet points:

- **Build a model using the available test data to classify the remaining useful lifetime (RUL) for each component.**
- **Identify relevant test variables of the component based on historic test data (feature selection).**

4.3.1 Classification

There are several data mining techniques that can be used to predict a target variable. Some examples of these techniques are clustering, association rules and sequence analysis. Classification technique is a supervised machine learning technique, capable of processing a wider variety of data than regression and is growing in popularity (Phyu, 2009). Supervised meaning that the value of the target variable is known before hand and is used to build the model. Because the goal of this project is to predict the remaining useful lifetime of an component and the unit by which this is measured is the amount of flying hours, I have to divide the flying hours in groups (or classes). The classification algorithm can then find relationships between the values of the predictors (test data) and the values of the target (remaining flying hours). Different classification algorithms use different techniques for finding relationships. These relationships are summarized in a model, which can then be applied to a different data set in which the class assignments are unknown.

The reason I do this is because we still lack information on the distribution of the RUL values in the data set. It is therefore a logical first step to divide this into classes. The simplest method is a binary classification of the data. Here, the data set is split into two groups based on a chosen value (e.g. 4000 flying hours). The data instances that score lower than the chosen value are assigned a 0, otherwise a 1. It is also possible to create several classes. An interesting point to investigate is what the optimal number of classes are, in which the model scores best. Figure 4.1 shows a schematic overview of the classification process.



Figure 4.1: A schematic overview of the classification process.

4.3.2 Performance measures

After doing the feature selection, and of course, implementing the model and getting the output in forms of a probability to a class, the next step is to find out how effective the model is based on some metric using test data sets. Different performance metrics are used to evaluate different machine learning algorithms. Choice of metrics influences how the performance of machine learning algorithms is measured and compared. For classification models I make use of the well-known confusion matrix. This matrix provides us with the accuracy, precision and recall of the model. Other performance metrics are: log-Loss, accuracy, AUC (Area Under Curve) etc. For this project, the confusion matrix metrics are used to rank and compare the different models with each other. Before looking at the importance of each metric to our model, it is important to note the differences between the two classifying errors that can occur when the model makes a wrong prediction:

- **False Positives (FP):** False positives are the cases when the actual class of the data point was 0 (False) and the predicted is 1 (True). False is because the model has predicted incorrectly and positive because the class predicted was a positive one.
- **False Negatives (FN):** False negatives are the cases when the actual class of the data point was 1 (True) and the predicted is 0 (False). False is because the model has predicted incorrectly and negative because the class predicted was a negative one.

In our model, a false positive would mean that a component that did not reach a certain threshold (in flying hours) was wrongly classified in a class above this threshold. This means the model overestimated the condition or health of the component. And a false negative would mean the opposite (the model underestimates the condition of the component) The associated penalty or costs are not the same for both errors as one would imagine. A false positive would mean that a component that was going to have an early failure was not detected by the model. This would not have a major implication, but is still undesirable. A false negative would mean that the model predicts an early failure, while this is not the case. Such an outcome would require maintenance personnel to follow the model and perform (unnecessary) maintenance action. Therefore, minimization of false negatives has a higher priority than minimizing false positives. This can be realized by putting an extra penalty on false negatives mis-classifications. This is discussed in detail in Section 7.6.

Some of the performance metrics that are associated with the confusion matrix were mentioned earlier. Using only the model's accuracy would not be suitable for this project. This is because accuracy is a good measure when the target variable classes in the data are nearly balanced, which is not the case with the component data. Therefore the models will be evaluated according to their precision and recall values as well. Precision is about being precise. So even if the model managed to only predict one component instance as being below the threshold, and it captured it correctly, then the model would be 100% precise. Recall (also called positive predictive value) is not so much about capturing cases correctly but more about capturing all cases that are below that threshold correctly. So if the model simply always says; every case is "below the threshold", it would have 100% recall.

In our case, I want to focus more on minimizing False Negatives. That means I would want our Recall to be as close to 100% as possible without precision being too bad. A metric that combines

both metric to one value is called the F1-score. This performance measure will be used for the final models. The expressions used to calculate the precision (p), recall (r) and F1- score are:

$$p = \frac{TP}{TP + FN}$$
$$r = \frac{TP}{TP + FP}$$
$$F1 - score = 2 \cdot \frac{p \cdot r}{p + r}$$

4.3.3 Initial Assessment Tools and Techniques

The main risk for this project is that the needed data is unavailable or insufficient. The contingency for this risk of unavailable data is that the recommendation can be made on what data should be acquired to perform the data mining techniques. If, for example, it turns out that the data is insufficient, then alternative data sources can be looked up to supplement the set. This is done in accordance with the project team. A shortage of historical test data can lead to a less than suitable predictive model (in many cases). It may have missing values that will skew the results and/or suggest unrepresentative results due to working with data that is not a representative sample of the population. This can cause even more challenges in data usefulness, predictive accuracy and variable interaction effects. However, limited data does not have to be a major roadblock in the modeling process. Because there are a number of data mining techniques that deal with this challenge.

The automation process in which test data records from historical component work orders are converted to horizontal excel rows has already been done. A TSR (Time Since Repair) value is given for each work order. This value indicates how long that component was in the field, before being sent back for repair. It may happen that one work order has different test data records. This is because some component's have multiple test runs before they are checked out. This can be problematic for determining the correct test data. This challenge will be tackled in the next chapter.

The software system with which the data streams are integrated is called **Alteryx**. This is a data analytics platform used to combine large streams of raw data from different sources, and apply data science techniques in order to get useful insights. **Alteryx** is compatible with a variety of other platforms and programming languages, like: **tableau**, **R** and **Python**. As a result, there is great freedom of choice in choosing the right programming language for the model.

Chapter 5

Data Understanding

This chapter is the result of the second phase of the CRISP-DM methodology. This has been executed as described in the methodology section 3.4.2. The objective of the data understanding phase is to discover which data is available, to get familiar with this data, to examine the quality of the data, and detect interesting characteristics in the data.

5.1 Data Collection

5.1.1 Data Sources

Collecting the data was not part of the objectives of this project. A data set containing test data records from component test runs was already available. These test data records stem from historic work orders that have been stored in the Fokker Services data base. On the test stands of the component shop located at the Fokker Schiphol site, several tests are performed on a daily basis to evaluate the condition of the components. Here, the test values of the component are measured and it is checked whether they fall within the permissible limit values required by flight regulations. Appendix E.3 shows an example of these test limits for the slip test. If this is the case for all tests, the component is allowed to be shipped back to the client. The acquired data is then linked to the work order and serial number of the specific component to create a unique test data record. There are a total of 18 tests for the component that have a total duration of approximately 4 to 8 hours depending on the amount of fails during the run. A list of the first 9 steps of the component test procedure, with a short description per step is shown in Table 6.1. The procedure and conditions of these tests are stipulated in a detailed component maintenance manual (CMM).

Other sources of data are the ERP system of Fokker Services called PENTAGON. This system keeps track of the components used during the MRO activities of each work order. It handles all materials management, supply chain management, trace-ability, maintenance and quality assurance. Additionally, the software also supports outside repairs, exchanges, consigned inventory and lot purchases. It also interfaces with leading third-party networks and services such as ILS, Aviall, SPEC2000 and AeroXchange. Information about the component type (mechanical, electrical or hydraulic), size and cost are found in an excel file called the Part Number (PN) master. PENTAGON also stores the amount of working hours and lists the employees that worked on a specific work order. Employees are identified using a unique employee-id, this is done for confidentiality reasons.

5.1.2 Collection Procedure

The test data was collected using an SQL query. SQL is a domain-specific language used in programming and designed for managing data held in a relational database management system (RDBMS). It is particularly useful in handling structured data where there are relations between different entities/variables of the data. The query generates an excel file by reading the test data

	Test	Description
STEP 1:	Resistance Check	Resistance check for the governor servo valve, permanent magnet generator, exciter field and main stator. Test may be performed before mounting the component on the test stand.
STEP 2:	Insulation Resistance Check	Voltage test at uniform rate of 250 to 500 volts per second.
STEP 3:	Permanent Magnet Generator Voltage Check	Permanent magnet generator voltage adjustment and check by applying a 3-phase, wye-connected resistive load.
STEP 4:	Open Circuit Saturation Check	Circuit saturation curve check by adjusting the voltage power supply
STEP 5:	Current Transformer Phasing Check	Optional CT current check
STEP 6:	Optional Patch Filter Check	A check of the path filter on chunks of metal and other nonmagnetic particles
STEP 7:	Slip Test	A slip test with a load of 3kW, by adjusting the input speed and load steps to 40kW
STEP 8:	a) Generation Rated Load Test	Generator load test, can be performed either Line-to-Neutral Voltage 120 or 115 V
	b) Generator Two Per Unit Load Test	Same as STEP 8, but increasing the load in one step to 60 kVA and decreasing the load to 40kVA within 5 seconds
STEP 9:	Short Circuit Check	Short circuit test by applying a 3-phase symmetrical circuit for a maximum of 5 seconds

Table 5.1: component test stand procedure. Source: Component Maintenance Manual (CMM)

records and systematically adding them to an Excel file. See Figure 5.1 for a schematic overview of this process. Each test data record contains a work order number and a serial number, based on these identification the test values are added. Several test data records can exist for a single work order. In that case, they are added in chronological order. The next step in the query is to add the used parts, the number of hours worked and the employees who have worked on them as separate columns in the file. Finally, I look at the moment the component returns to the shop for the first time. The amount of flying hours the component made since its last visit is called the time-since-repair (TSR) value of the component. This is not a indicator of physical time spent in the field, but more about the actual usage of the generator in terms of flying hours in an aircraft. If the component has comes in for an overhaul (complete repair), this is indicated as time-since-overhaul (TSO). The final data sets contains test data records up to 2016, more recent data is not usable, because these components are still in the field and the TSR/TSO values are therefore not yet known. During the collection of the data, a problem arose when extracting data from 2010 onward, this was due to a bug in the source code of the software system with which the data records can be read. This was remedied in time, by manually extracting the data.

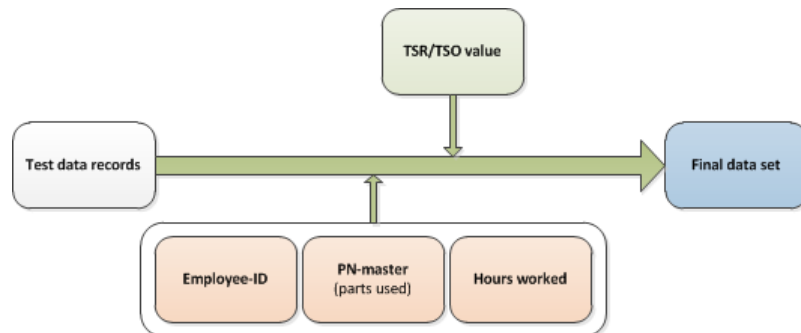


Figure 5.1: A schematic overview of the data collection procedure.

5.2 Data Description

This section contains a descriptive analysis of the final data set. This analysis is split into a summary of the data, followed by a brief introduction to the insights found in the data. This

Feature type	Variable count	Class type
Test data readings	212	numeric
Administrative data	15	composite
Client data	3	composite
Maintenance performed	2	boolean
Hours worked	1	numeric
Parts used	1	string
Identifiers	5	composite

Table 5.2: Data set summary

analysis has been conducted using various tools and software; Python, R, Excel and Tableau.

5.2.1 Summary

The size of the final data set was 398 test data records of 278 work orders, and an initial count of 268 features. Table 6.2 gives an overview of these features. The bulk of the input data are numerical test data readings, but the data also includes other relevant features like: account-number (this specifies which client uses the component), warranty description, type of inspection (repair/overhaul), hours worked, parts used and employee and serial ID's.

5.2.2 Initial Insights

The first thing I notice when looking at the data, is the amount of missing values in the test readings. Data can have missing values for a number of reasons such as observations that were not recorded and data corruption. Handling missing data is important as many machine learning algorithms do not support data with missing values. Fortunately there are a number of techniques in Python that can be used to handle and impute missing values, more on this in Chapter 7.

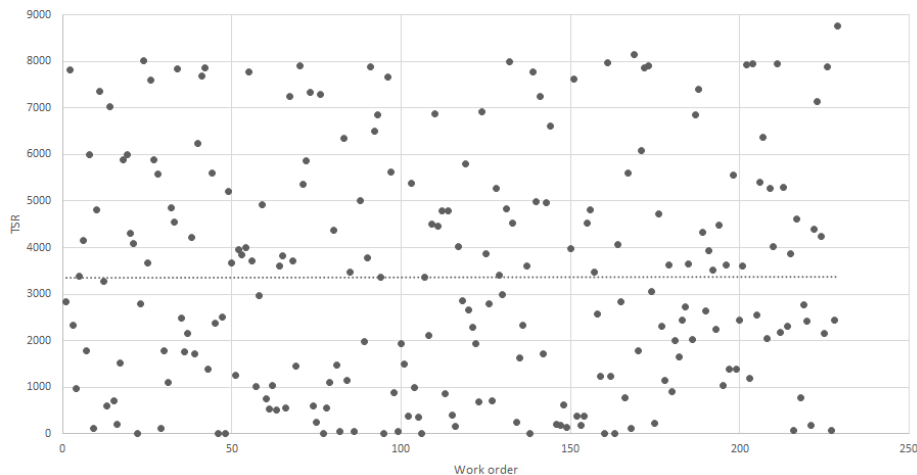
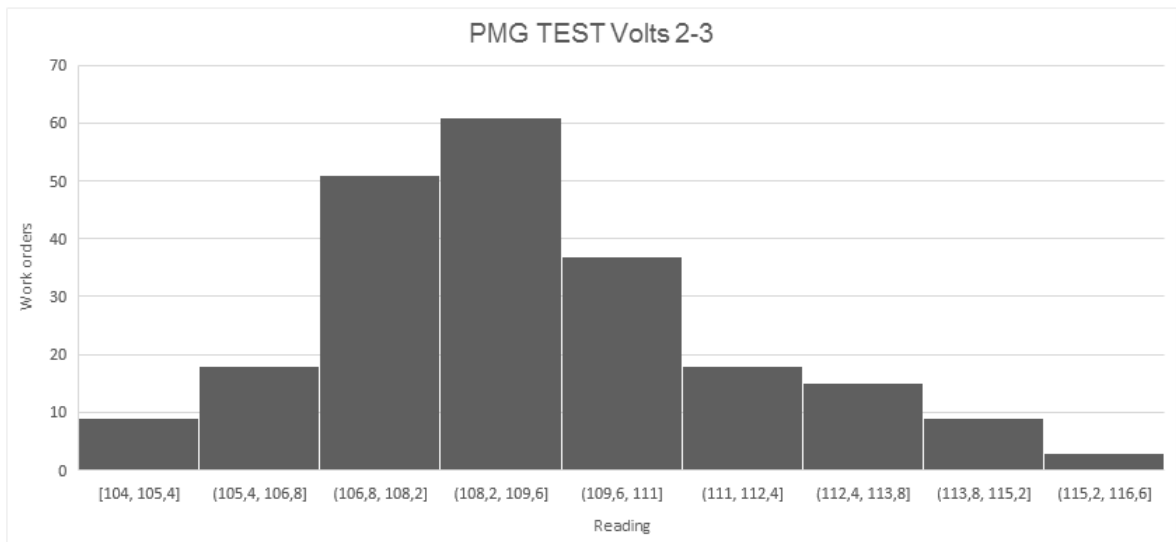


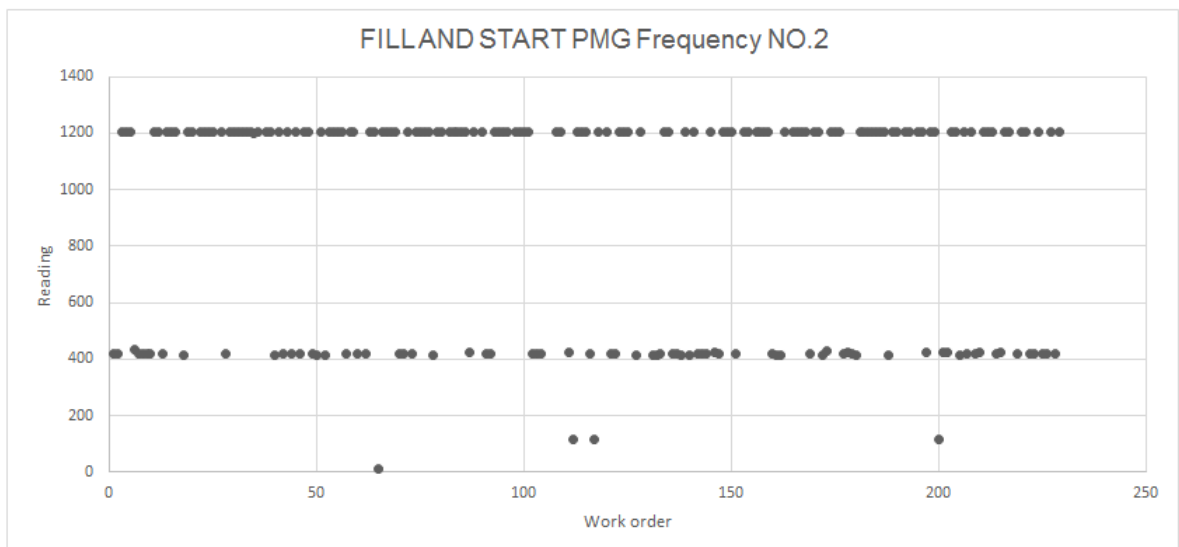
Figure 5.2: A scatter-plot of the TSR values in the final test data. (*mean = 3200 flying hours*)

Another interesting point is that the values of TSR are spread between 0 and 8000 flying hours with a slight skew at 8000 hours limit. This is correct with the initial observation that many components are sent back prematurely. Ideally, a large majority of components would have a TSR value of 6000 flying hours or more. The scatter plot of the TSR values in Figure 5.2 shows this sparsity and the early failures well. The figure clearly shows that many components are experiencing an early failure. The average TSR values is at 3200 flying hours.

It is also clear that many of the variables in the final test data are not relevant for the prediction of the TSR values. These variables will be removed at the data cleaning step. Examples of these variables are; serial number and several customer reference numbers. Another observation is that the test data readings follow different distributions. Of the 212 variables, 168 are variables have a normal distribution, and 44 variables assume 1 or 2 specific values. See Figure 5.3 for an example of both types. The readings with a normal distribution can be explained by continuous physical quantities that can deviate naturally from a certain average. The other type variables, which only have 1 to 2 (in rare cases 3) values, are usually frequency readings.



(a) Test reading with a continuous normal distribution



(b) Frequency reading

Figure 5.3: Two examples of different variable types in the data set

5.3 Data Quality

Data quality refers to the condition of a data set containing a set of values of qualitative or quantitative variables. There are many definitions of data quality but data is generally considered high quality if it is "fit for (its) intended uses in operations, decision making and planning" (Thomas and Redman 2008). Alternatively, data is deemed of high quality if it correctly represents the real-world construct to which it refers. Furthermore, as data volume increases, the question of internal data consistency becomes significant. For this section I will focus on the following key data quality components:

- **Accuracy:** Degree at which the data actually represents the real world status that it is measuring. May be calculated by using an automated method with the help of various lists and mapping.
- **Completeness:** Level at which desired data attributes are supplied.
- **Timeliness** (age of data): Extent to which data is adequately updated for a current venture.

5.3.1 Accuracy

The measurements from the test data records are very accurate. This is because the regulations and conditions of the tests have been precisely formulated in the CMM. Also, the test equipment that is used for the measurements have a high standard. This is also due to the strict regulations and safety standards in the aviation industry. As a result, the most minor deviations from the norm are measured and recorded. This is also important, because for some variables a deviation of 0.05 Volts can be a reason to reject a component.

The other variables, like employees and hours worked are also accurate. These have been manually integrated in the data set using sources that register the information based on the test reports from engineers. These reports also contain the amount of work spent, and the number and type of components used. This data has been checked for errors before being integrated in the final data set.

5.3.2 Completeness

Data completeness is an important quality component as it improves and supports the function of predictive models. There are two questions that arise when assessing data completeness for this project: "how complete is the current data set" and "is the provided data sufficiently complete for the use of prognostics on the output variable? To give an answer on the first question, I made an analyses on the amount of missing values throughout the data set. Most missing values originate from test data readings. The main cause for this is that the reading has not been performed because it was not deemed necessary due the result of a previous test. Table 5.3 gives an overview of the total amount of missing values from the sequential test runs. Each subsequent test run covers less attributes than the preceding one because they are tested again for only a sub set of the attributes. This can be illustrated as follows: if a component has passed all tests except one, it will be rejected. After performing repairs or replacing the defect components, the component must be put on the test stand again. In this case however, only the mandatory tests are performed. The readings for the non-mandatory (or optional) tests are left out, and are therefore marked as missing values in the data set. Some components have to be placed on the test stand for more than six times. The number of relevant tests decreases as the component needs to be run on the stand more often. The average number of missing values for the sixth test run is 87.8%.

5.3.3 Timeliness

In the field of data science, timeliness and accuracy are fundamental prerequisites when it comes to performing data analysis. The final data set for this project contain test data records from as

Test run	Count	Missing values*	Percentage
1st (initial) run	63	2	0.009
2nd run	52	22	0.096
3rd run	63	68	0.296
4th run	47	131	0.570
5th run	19	183	0.796
6th run	1	202	0.878

Table 5.3: Summary of missing values. **Average count of missing values per test variable.*

early as; 17 March 2010, until 26 November 2013. This is not the most recent data available for component units, but it is the only data available of which the TSR (output variable) is known. The data is considered reliable for analysis purposes because the test procedure and the contents of the component have remained unchanged the last years.

Chapter 6

Data Preparation

This chapter explains the steps taken during the data cleaning steps performed on the acquired data set. The output of this phase are the train and test sets that are used for the modeling phase. I must emphasize that the process of data cleaning is not sequential in nature. Between each of the different modeling techniques, iterative data transformation actions have been committed to make the data suitable for the model in question. These steps are clearly described in the next chapter. For the cleaning steps I used primarily the known Python libraries: `scikitlearn`, `numpy`, and `pandas`. This chapter is divided in three steps, based on the KDD process framework. These steps are feature selection, data cleaning, and data integration.

6.1 Feature Selection

Data selection is defined as the process where data relevant to the analysis is decided and retrieved from the data collection. One of the goals of this project is to design a complete Python package, in which the data is automatically selected and processed for analysis. The removal of irrelevant columns from the data set was done by making use of expert knowledge. The columns that I dropped in were:

- **Non-numeric variables.** Columns containing descriptions or references
- **Duplicate columns.** Some test files contained multiple readings from the same run.
- **Non-explanatory variables.** These are customer of order identifiers, account reference numbers, etc.
- **Empty columns**

6.2 Data Cleaning

The complete data cleaning script written in Python 3 can be seen in Appendix C. What follows is a step-by-step explanation and rationale of the code.

6.2.1 Work order duplicates

After importing the data set (`df`) and relevant packages, the first step was to remove the duplicates in the work orders. The fact is that for each run file a separate work order is created and components that have to go over test bench several times will generate a doubling in the test results. I solved this by taking the most recent (or most recently executed) test from every run file. This makes it certain that only the approved test values are included:

```
df = df.drop_duplicates(subset = 'HDR.USER_DOC', keep = 'last')
```


6.2.2 Missing values

For this part I created a separate data-frame for the sensor data: `df_sensor`. After inspecting the sensory data, it became clear that there were a lot of empty values. Apparently this is a normal phenomenon during test runs. For example, some load checks are skipped for time saving purposes if there is sufficient evidence that the component will succeed the test. In Python these missing values are indicated with `NaN`. I handle the missing data with the following code:

```
from sklearn.preprocessing import Imputer
imputer = Imputer(missing_values = 'NaN', strategy = 'mean', axis = 0)
imputer = imputer.fit(df_sensor)
```

```
X[:, 1:3] = imputer.transform(df_sensor)
```

The mean imputer calculates the mean value of the column and replaces the `NaN` values with the mean value. Some columns have a high number of missing values. I set a threshold of 40% on the amount of missing values that are acceptable. Columns that have a higher percentage of missing values are dropped:

```
df_sensor['HDR.USER_DOC'] = df['HDR.USER_DOC']
thresh = len(df_sensor)*.4
df_sensor = df_sensor.loc[:, pd.notnull(df_sensor).sum()>thresh]
```

For the parts (`df_parts`) and employees (`df_employee`) data I used a different approach. These variables are binary variables that also contain missing cells. These are simply filled using the `fillna` function:

```
df_employees = df.loc[:, 'Employee_is_1': 'Employee_is_17']
df_employees.fillna(value=0, inplace=True)

df_parts = df.loc[:, 'PN_is_58163-8': 'PN_is_MS9276-09']
df_parts.fillna(value=0, inplace=True)
df_parts['Total_Parts'] = df_parts[list(df_parts.columns)].sum(axis=1)
```

6.2.3 Encoding of categorical variables

The data set contains a number of categorical variables that must be transformed into numeric arrays. An example of a categorical variable is `QFINISH.WORK_PERFORMED`. This variable consists of the following unique values: `REP` (repair), `OVH` (overhaul) and `INS` (inspection). To create binary dummy variables for each value, I use the `OneHotEncoder` function of scikitlearn on the subset of categorical variables; `df_categorical`. This feature represents categorical variables as binary vectors. This first requires that the categorical values be mapped to integer values. Then, each integer value is represented as a binary vector that is all zero values except the index of the integer, which is marked with a 1:

```
from sklearn.preprocessing import OneHotEncoder

onehot_encoder = OneHotEncoder(sparse=False)
df_categorical = onehot_encoder.fit_transform(df_categorical)
```

The same procedures was applied to the employee and part-number data frames. Some parts were removed from the unique value list however. This is because a number of parts belong to a category that is always used for repairs and does not provide any additional information about the type of repair. Examples are; bearings, plastic isolation, etc.

6.3 Data Integration

The final step was to merge the different data-frames after pre-processing into the final data set. And divide the data into a train and test set. To make the column names more readable, I removed the punctuation marks and unwanted characters from the column labels:

```
def remove_invalid_chars(df):
    df.columns = [elem.replace(",","").replace("[","").replace("]",
    '').replace('<', '') for elem in df.columns.values]

remove_invalid_chars(df)
```

I split the output value (TSR) into three different bins, based on the that will serve as output classes. Each bin specifies the limits for the TSR values of the component work orders. The values in these bins are therefore binary and indicate whether or not the component falls within these limits.

Example: a work order of a component returning with a TSR value of 5,500 flight hours for overhaul, gets a value of 1 for the first 2 bins, and a value of 0 for the last bin (TSR between 6,000 and 8,000 flight hours).

	TSR_2000	TSR_4000	TSR_6000
count	227.000000	227.000000	227.000000
mean	0.647577	0.387665	0.176211
std	0.478781	0.488294	0.381842
min	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000
50%	1.000000	0.000000	0.000000
75%	1.000000	1.000000	0.000000
max	1.000000	1.000000	1.000000

Figure 6.1: Summary statistics of the target variable: TSR

Merging all the input data together results in a cleaned data set with 227 rows and 482 columns or features, with no missing values. Finally, splitting the data results in a train set of 127 rows and a test set of 100 rows. A distribution of 60% train data and 40% test data is often used in machine learning settings with less than 200 instances (Adi Bronchstein, 2017). I have chosen a test set of 44% to get a nice round amount of 100 instances in the test set. During some modeling techniques, I use the k-fold cross validation method, whereby the data set is divided into 5 equal sets. The code below shows the train/test split in Python. To specify the input data range I use the `df.loc` function with the column names.

```
test_size = .44
seed = 4

train, test = train_test_split(df, test_size=test_size, random_state=seed)

Y_train = train['TSR']
X_train = train[list(df.loc[:, 'Employee_is_17':
'SHORT_CIRCUIT_TEST-Gen_Short_Circuit_Current'])]

Y_test = test['TSR']
```

```
X_test = test[list(df.loc[:, 'Employee_is_17':  
'SHORT_CIRCUIT_TEST-Gen_Short_Circuit_Current'])]
```

Chapter 7

Modeling

This chapter is the result of the fourth phase of the CRISP-DM methodology, see section 3.4.4. While most people think that modeling is the most time consuming part of a project, more time was spent on understanding the data and preparing it then on the modeling part itself. First, an overview of the modeling assumptions is presented. This is followed by the test design of the data analysis. From section 7.4 to 7.6 I give a theoretical description of the different data mining techniques that were used during the project, and their respective parameters. Finally an assessment and validation of the generated models is given, with an overview of the revised parameter settings.

7.1 Modeling Assumptions

The choice of data that one includes in the model, how the data are processed, and the algorithms themselves all rely on or create assumptions about the underlying data structure. Assumptions are made at every point of the modeling process. This is why “data-driven” approaches and unsupervised learning methods should be validated against both theoretical expert-knowledge and multiple tests of model quality.

There are two basic assumptions underlying the machine learning techniques that I will use.

- Firstly, the available data instances are independent and identically distributed (IID), according to an unknown probability distribution. Models are trained on a set and tested on another, under the assumption that the two are very strongly correlated (once overfitting is taken care of).
- Secondly, the algorithms do not care about the order in which data is presented, and all data is treated the same.

7.2 Model Selection

In section 3.4.4 I already stated that the three candidate models I will use for this project are: Neural Networks, Gradient Boost trees and Forest Classification trees. Often the hardest and crucial part of solving a machine learning problem is finding the right estimator for the job. Different estimators are better suited for different types of data and different problems. A very useful flowchart is designed by scikit-learn to give users a bit of a rough guide on how to approach problems with regard to which estimators to try on the data. The complete chart can be found in Appendix D. Following this guide, I made a list of candidate models that are applicable for the classification problem at hand. Keeping the project duration in mind, I decided to focus on three modeling techniques. This decision was made by comparing the models on different modeling criteria, see Table 7.1.

	KNN	Logistic regression	Naive Bayes	Decision trees	Random Forests	XGBoost	Neural networks
Problem Type	Either	Classification	Classification	Either	Either	Either	Either
Results interpretable by user?	Yes	Somewhat	Somewhat	Somewhat	A little	A little	No
Easy to explain algorithm?	Yes	Somewhat	Somewhat	Somewhat	No	No	No
Average predictive accuracy	Lower	Lower	Lower	Lower	Higher	Higher	Higher
Training speed	Fast	Fast	Fast	Fast	Slow	Slow	Slow
Prediction speed	Fast	Fast	Fast	Fast	Moderate	Fast	Fast
Amount of parameter tuning needed	Minimal	None	Somewhat	Some	Some	Some	Lots
Perform well with small number of observations?	No	Yes	No	No	No	No	No
Handles lots of irrelevant features well?	No	No	No	No	Yes	Yes	Yes
Automatically learns feature interactions?	No	No	No	Yes	Yes	Yes	Yes
Gives calibrated probabilities of class membership?	Yes	Yes	No	Possibly	Possibly	Possibly	Possibly
Parametric?	No	Yes	Yes	No	No	No	No
Features might need scaling?	Yes	No	No	No	No	No	Yes

Table 7.1: Comparing Supervised Learning Algorithms

From the table we can see that Neural Networks and the ensembling tree techniques have the best overall score (dark gray corresponds to more important criteria). I decided to include a logistic regression model. The reason for this is to include a model that is robust and does not decrease in predictive performance when there are fewer number of observations.

7.3 Test Design

In order to build the models and evaluate their performance I divide the final data set randomly in a training set and a test set. In order to avoid underfitting and overfitting I make use of the k-fold Cross validation method. The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation. Cross-validation (CV) is primarily used in applied machine learning to estimate the accuracy of a machine learning model on unseen data. That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model. The following procedure is followed for each of the k “folds”: A model is trained using $k - 1$ of the folds as training data. Then the resulting model is validated on the remaining fold of the data (it is used as a test set to compute a performance measure such as accuracy).

```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(model, df_input, TSR_4000, cv=5)
scores
```

Figure 7.1: Test Design using CV

The performance measure reported by k -fold cross-validation is then the average of the values computed in the loop. This approach can be computationally expensive, but does not waste too much data (as is the case when fixing an arbitrary validation set), which is a major advantage in problems such as inverse inference where the number of samples is very small. The value of k must be chosen carefully to prevent a misrepresentation of the model accuracy, such as a high bias or high variance. For this project I choose a value of $k = 10$. This is because the number of samples in the final data set is limited and it has been found through experimentation to generally result in a model evaluation with low bias and a modest variance (Kuhn and Johnson 2013). The

Modeling notation	Python code	Definition
$X := \{x_1, \dots, x_n\}$	df_final	Matrix containing all input data
$y \in N$	TSR	Data column containing the TSR value
$y_{2000} \in \{0, 1\}$	TSR_2000	Binary classification for $TSR = 2000$
$y_{4000} \in \{0, 1\}$	TSR_4000	Binary classification for $TSR = 4000$
$y_{6000} \in \{0, 1\}$	TSR_6000	Binary classification for $TSR = 6000$
\hat{y}_i for $i \in \{2000, 4000, 8000\}$	TSR.predict	TSR-class probability predictions

Table 7.2: Definition of the model input and output

simplest way to use cross-validation is to call the `cross_val_score` function in Python. Below you can see the code used for the cross-validation split used to evaluation all the models.

The first data mining goal of this project is to classify the component into classes based on a prediction of the RUL using the available test data. This classification problem can be solved by numerous classification algorithms. Some of which are more complex than others. Together with the project team I decided to create several prediction models, using the following techniques; Neural Networks, Gradient Boost Trees (GBT), Forest Classification Trees (FCT) and Logistic Regression. The models are then compared with each other using the performance indicators discussed in section 4.3.2.

In the next sections, I give a description of each modeling technique along with a step-by-step explanation of the Python code I wrote to build the prediction models. For each modeling approach I use the same definition for output and input, see Table 7.2. The derivation of the binary output variables is explained in Chapter 6.

7.4 Neural Network

A supervised neural network, at the highest and simplest abstract representation, can be presented as a black box with 2 methods learning and predicting. The learning process takes the inputs and the desired outputs and updates its internal state accordingly, so the calculated output gets as close as possible to the desired output. The predict process takes the input and generates the most likely output according to its past “training experience”. The training of a neural network is an iteration process where the internal state is updated at each iteration. The error at each iteration is reduced by adapting the weights of the model to each input variable, thus creating a weighted sum of all the input variables using a bias. A neural network is very suitable for recognizing non-linear relationships between the input and the output.

A typical neural network has anything from a few dozen to hundreds, thousands, or even millions of artificial neurons called units arranged in a series of layers, each of which connects to the layers on either side. The input units receive input data from the outside world that the network will attempt to ‘learn’ about. Other units sit on the opposite side of the network and signal how it responds to the information it’s learned; those are known as output units. In between the input units and output units are one or more layers of hidden units, which form the majority of the artificial brain. The connections between one unit and another are represented by a number called a weight which can be either positive or negative. The higher the weight, the more influence one unit has on another. Information flows through a neural network in two ways. Patterns of information are fed into the network via the input units, which trigger the layers of hidden units, and these in turn arrive at the output units. This common design is called a feed-forward network. Each unit receives inputs from the units to its left, and the inputs are multiplied by the weights of the connections they travel along. Every unit adds up all the inputs it receives in this way and if the sum is more than a certain threshold value, the unit “fires” and triggers the units it’s

connected to (those on its right). Figure 7.2 a Neural Network with five inputs, three hidden layers and one output.

For a neural network to learn, it uses an element of feedback process called back-propagation. This involves comparing the output a network produces with the output it was meant to produce, and using the difference between them to modify the weights of the connections between the units in the network. In time, back-propagation causes the network to learn, reducing the difference between actual and intended output to the point where the two exactly coincide.

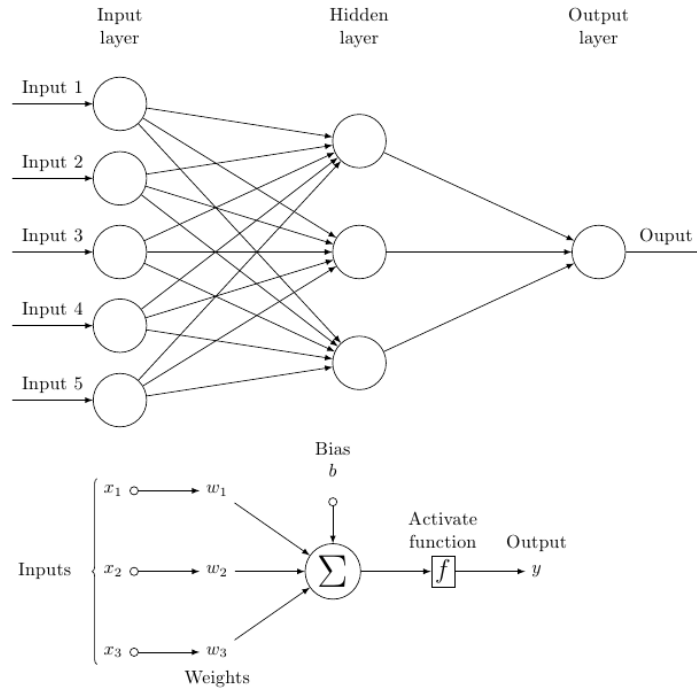


Figure 7.2: Neural Network diagram

To summarize, a Neural Networks consist of the following components:

- An input layer, x
- An arbitrary amount of hidden layers
- A set of weights and biases between each layer, W and b
- A choice of activation functions for each hidden layer, σ

The first step in building our Neural Network, is defining the model class in `Python`, its inputs and outputs, and its initial weight parameters, see Figure 7.3 It is common practice to use only one hidden layer, because adding a second or third layer does rarely improve the performance of the mode, but it does increase its complexity considerably.

Deciding the number of neurons in our hidden layer is a very important part of deciding the overall neural network architecture. Though these layers do not directly interact with the external environment, they have a tremendous influence on the final output. Using too few neurons in the hidden layer will result in something called underfitting. Underfitting occurs when there are too few neurons in the hidden layers to adequately detect the signals in a complicated data set. Using

too many neurons in the hidden layers can result in several problems. First, too many neurons in the hidden layers may result in overfitting. Overfitting occurs when the neural network has so much information processing capacity that the limited amount of information contained in the training set is not enough to train all of the neurons in the hidden layers. A second problem can occur even when the training data is sufficient. An inordinately large number of neurons in the hidden layers can increase the time it takes to train the network. The amount of training time can increase to the point that it is impossible to adequately train the neural network. There are many rule-of-thumb methods for determining the correct number of neurons to use in the hidden layer: The number of hidden neurons should be

- In the range between the size of the input layer and the size of the output layer
- 2/3 of the input layer size, plus the size of the output layer
- Less than twice the input layer size

My choice for the number of hidden neurons is based on a review article by Panchal and Panchal (2014), where they came up with the following equation:

$$N_h = \frac{(\alpha \cdot (N_i + N_o))}{N_s}$$

Where, N_h is the number of hidden neurons, N_i is the number of input neurons (in our case 120 input neurons), α is an arbitrary scaling factor (usually 2-10), N_o is the number of output neurons (in our case that is 1 output neuron). And N_s is the number of samples in the data set used to train the network (in our case this is 80% of 230; 184). Therefore the number of hidden neurons is set on an initial number of 6 hidden neurons ($\alpha=8$).

```
class NeuralNetwork:
    def __init__(self, x, y):
        self.input = x
        self.weights1 = np.random.rand(self.input.shape[1],6)
        self.weights2 = np.random.rand(6,1)
        self.y = y
        self.output = np.zeros(y.shape)
```

Figure 7.3: Initialization of the Neural Network

The output of a 6-layer Neural Network with a set of parameters (θ) is calculated using the weights W and the biases b . At this stage I have the initial probabilities of the output and the actual output class. The model then "adapts" the weights in order to minimize the error between the predicted value \hat{y}_i and the actual value of TSR y_i , using a so-called; loss function. I set the loss function to be the sum of squares of the absolute errors (which is the most commonly used loss function in machine learning):

$$L(\theta) = \sum_i (y_i - \hat{y}_i)^2$$

The process of minimizing the the loss function (back-propagation), has a number of parameters that play a role in this process. Some parameters are left at their default values (batch size and solver algorithm for example). Other parameters are fine tuned to improve the back-propagation process, keeping the computation time in mind. After the training process, the model is then used to predict the class probabilities c for the test set. The output for each work order (sample) is an array with three probabilities, each corresponding to the predefined TSR class. The final output value used for model evaluation is simply obtained by choosing the class with the highest probability:

$$\hat{y}_i = \operatorname{argmax}_{j \in \{2000, 4000, 8000\}} P(c_j | x_i)$$

These output probabilities are obtained by the model, using the softmax function. This function is a generalization of the logistic function that “squashes” a K -dimensional vector z of arbitrary real values to a K -dimensional vector $\sigma(z)$ of real values in the range $[0, 1]$ that add up to 1. A consequence of using the softmax function is that the probability for a class is not independent from the other class probabilities. Appendix C.1 shows the complete `Python` script for the NN model.

7.5 Decision Trees

The main causes of difference in actual and predicted values in any machine learning technique, are noise, variance, and bias. Decision trees use a technique called ensemble, that helps to reduce these factors (except noise, which is irreducible error). A decision tree ensemble is just a collection of predictors, or CARTs (classification and regression trees) which come together (e.g. mean of all predictions) to give a final prediction. A CART is a bit different from decision trees, in which the leaf only contains decision values. In CART, a real score is associated with each of the leaves, which gives us richer interpretations that go beyond classification. Ensembling techniques are further classified into Bagging and Boosting.

- Bagging is a simple ensembling technique in which I build many independent predictors/-models/learners and combine them using some model averaging techniques. (e.g. weighted average, majority vote or normal average). I take random sub-sample/bootstrap of data for each model, so that all the models are little different from each other. Each observation is chosen with replacement to be used as input for each of the model. So, each model will have different observations based on the bootstrap process. Because this technique takes many uncorrelated learners to make a final model, it reduces error by reducing variance. Example of bagging ensemble is Random Forest models.
- Boosting is an ensemble technique in which the predictors are not made independently, but sequentially. This technique employs the logic in which the subsequent predictors learn from the mistakes of the previous predictors. Therefore, the observations have an unequal probability of appearing in subsequent models and ones with the highest error appear most. (So the observations are not chosen based on the bootstrap process, but based on the error). The predictors can be chosen from a range of models like decision trees, regressors, classifiers etc. Because new predictors are learning from mistakes committed by previous predictors, it takes less time/iterations to reach close to actual predictions. Choosing the stopping criteria is important because a poor choice could lead to overfitting on training data. Gradient Boosting is an example of boosting algorithm. Boosting is a sequential technique which works on the principle of tree ensembles. It combines a set of weak learners and delivers improved prediction accuracy. At any instant t , the model outcomes are weighed based on the outcomes of previous instant $t - 1$. The outcomes predicted correctly are given a lower weight and the ones miss-classified are weighted higher.

Applying boosting algorithms is a good way of improving the accuracy of a predictive model, especially in combination with feature engineering. In this project, I used two widely used and very popular ensembling techniques; XGBoost and Random Forest. In section 7.6 I discuss the methods used for feature engineering.

7.5.1 XGBoost

XGBoost stands for “eXtreme Gradient Boosting”, where the term “Gradient Boosting” refers to the ensemble algorithm mentioned earlier. Gradient boosting is an approach where new models are created that predict the residuals or errors of prior models and then added together to make the final prediction. It is called gradient boosting because it uses a gradient descent algorithm to minimize the loss when adding new models. Both XGBoost and Random forest models use the

same ensemble algorithm, the only difference arises in the way in which the models are trained. This means that the models I write for XGBoost can also be used for the Random Forest model, and therefore repetition of the mathematical derivation is not needed.

As I mentioned previously, the prediction scores of each individual tree are summed up to get the final score. Mathematically, we can write our model in the form:

$$\hat{y}_i = \sum_K^{k=1} f_k(x_i), f_k \in F$$

Where K is the number of trees, f is a function in the functional space F , and F is the set of all possible CARTs. A salient characteristic of objective functions for decision trees is that they consist two parts: training loss $L(\theta)$ and the regularization term $\Omega(\theta)$. The loss function is exactly the same as the function used for the Neural Network model, see Section 7.2. The regularization term controls the complexity of the model, which helps us to avoid overfitting. A high regularization means that the model is complex and could potentially lead to overfitting and long running times. For a simple and predictive model, both $L(\theta)$ and $\Omega(\theta)$ have to be balanced (bias-variance trade-off). The objective function to be optimized is given by:

$$obj(\theta) = \sum_n^i l(y_i, \hat{y}_i) + \sum_K^{k=1} \Omega(f_k)$$

Before delving into the actual training process, I would like to highlight the XGBoost model parameters. Before running XGBoost, I have to set two types of parameters: booster parameters and task parameters. General parameters relate to which booster (tree or linear) and do not need to be set because they are automatically set by the XGBoost package in Python. The booster and task parameters are important because they define the environment of the learning process and they specify the learning task and the corresponding learning objective. See Figure 7.4 for the initial set-up of the XGBoost model. The parameters that are relevant for tuning are:

- **objective** (default=reg:linear). This parameter is the most important for the XGBoost model. It specifies which objective function is used for optimization. There are 14 different objective functions I can choose from, but the linear regression function which is the default is not the most suitable for our project. Here, I choose `multi:softmax`. This is an activation function that turns numeric outputs into class probabilities that sum to one. This is especially useful for this project, because we have defined three classes in the output of the model. The softmax function is calculated as follows:

$$S(y_i) = \frac{e^{y_i}}{\sum_i e^{y_i}}$$

- **eta** (default= 0.3, alias: `_rate`, range: (0,1)). Step size shrinkage used in update to prevent overfitting. After each boosting step, we can directly get the weights of new features, and eta shrinks the feature weights to make the boosting process more conservative.
- **max_depth** (default=6, range: (0,∞)). Maximum depth of a tree. Increasing this value will make the model more complex and more likely to over-fit.
- **tree_method** string (default=auto). This is the tree construction algorithm used in XGBoost. The choices are: `auto`, `exact`, `approx`, `hist`, `gpu_exact`, `gpu_hist`.
- **scale_pos_weight** (default=1). Controls the balance of positive and negative weights. I use this parameter to handle the unbalanced classes in the data set.

```

import xgboost as xgb
param = {
    'max_depth': 3, # the maximum depth of each tree
    'eta': 0.3, # the training step for each iteration
    'silent': 1, # logging mode - quiet
    'objective': 'multi:softmax', # error evaluation for multiclass training
    'num_class': 3} # the number of classes that exist in this dataset
    'num_round' = 20, # the number of training iterations
    'scale_pos_weight' : 1 #

XGBoost = xgb.train(param, dtrain)

preds = bst.predict(dtest)
best_preds = np.asarray([np.argmax(line) for line in preds])

```

Figure 7.4: Initial set-up of a XGBoost model in Python

Now the model is a partially specified equation: a numeric function of the input data, with some unspecified numeric parameters. During the training procedure, the model is searching for the 'best' set of parameters that optimize the specified objective function. The model needs to learn the functions f_i , each containing the structure of the tree and the leaf scores. Learning tree structure is much harder than the traditional optimization problem where you can simply take the gradient. It is intractable to learn all the trees at once. Instead, an additive strategy is used: the current classifier is fixed, and a new tree is added at a time. If we denote the prediction value at step t as \hat{y}_i^t . Then we have:

$$\begin{aligned}
 \hat{y}_i^{(0)} &= 0 \\
 \hat{y}_i^{(1)} &= f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i) \\
 \hat{y}_i^{(2)} &= f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i) \\
 &\dots \\
 \hat{y}_i^{(t)} &= \sum_{k=1}^t f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)
 \end{aligned}$$

Then we need to determine which tree to use for each step. The boosting algorithm adds the tree that optimizes the objective function the most. The objective function at time t is:

$$\begin{aligned}
 obj^t &= \sum_{i=1}^n l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^t \Omega(f_i) \\
 &= \sum_{i=1}^n l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) + \text{constant}
 \end{aligned}$$

Because we have the mean squared error as loss function, the objective becomes:

$$\begin{aligned}
 obj^t &= \sum_{i=1}^n (y_i - (\hat{y}_i^{(t-1)} + f_t(x_i)))^2 + \sum_{i=1}^t \Omega(f_i) \\
 &= \sum_{i=1}^n [2(\hat{y}_i^{(t-1)} - y_i)f_t(x_i) + f_t(x_i)^2] + \Omega(f_t) + \text{constant}
 \end{aligned}$$

This becomes our optimization goal for the new tree. The number of iterations during the training process is set automatically. The model will continue training until the validation score stops improving. The output of the XGBoost model can be represented by a ranking of the attributes in order of importance, and the actual decision tree. To plot the importance I use the

IPython function `xgboost.plot_importance(XGBoost)`. To plot the output tree via `matplotlib`, I use the function `xgboost.plot_tree()`, specifying the ordinal number of the target tree. This function requires `graphviz` and the `matplotlib` library. See Figure 7.5 for a plot of the baseline decision tree before parameter tuning. The final decision trees are presented in chapter 8 and Appendix E. The complete Python script for the XGBoost model can be found in Appendix C.2.

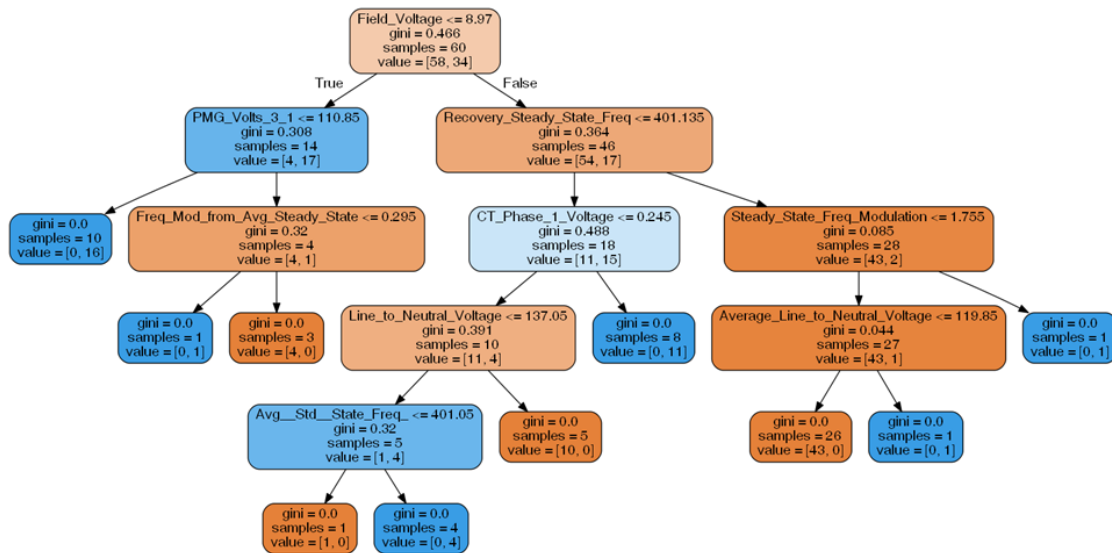


Figure 7.5: Plot of the XGBoost decision tree prior to parameter tuning

7.5.2 Random Forest

Another commonly used classifier is the Random Forest classifier (RFC). This technique is, like XGBoost, a tree-based model. It also makes use of a supervised algorithm and can be implemented in both classification and regression problems. RFC creates decision trees on randomly selected data samples, gets prediction from each tree and merges them together to get a more accurate and stable prediction. Figure 7.6 shows a representation of a RFC with two trees. The RFC algorithm works as follows:

1. Select random samples from the data (train) set
2. Construct a decision tree for each sample and get a prediction result from each decision tree.
3. Perform a vote for each predicted result.
4. Select the prediction result with the most votes as the final prediction.

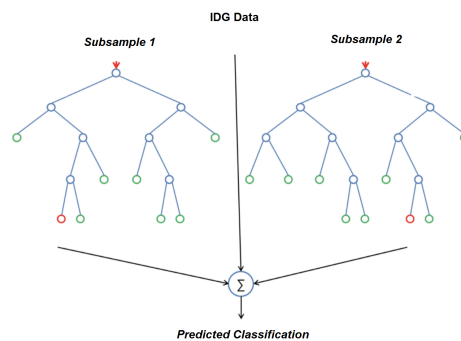


Figure 7.6: Random Forest Classifier with two trees

There are a lot of similarities between RFC and XGBoost models. The calculations that the models use for the fitting process are the same. Therefore I will not derive the mathematical calculation for the RFC here. The idea of bagging in random forest is very important. Bagging means Bootstrap aggregation. Bootstrap means generating random samples from the data set with replacement. The main reason of bagging is to reduce the variance of the model class. Like I mentioned earlier, XGBoost models also deal with the trade-off between variance and bias. The main difference between RFC and XGBoost is the method used to tackle this. Both RFC and GBM are ensemble methods, meaning they build a classifier out a big number of smaller classifiers. Now the fundamental difference lies on the method used:

Random Forest Classifier (RFC)

RFC uses decision trees, which are very prone to overfitting. In order to achieve higher accuracy, RF decides to create a large number of them based on bagging. The basic idea is to re-sample the data over and over and for each sample train a new classifier. Different classifiers over fit the data in a different way, and through voting those differences are averaged out.

Gradient Boost Machine (GBM)

GBM is a boosting method, which builds on weak classifiers (high bias, low variance). The idea is to add a classifier at a time, so that the next classifier is trained to improve the already trained ensemble. Notice that for RFC each iteration the classifier is trained independently from the rest.

When writing the RFC model, the Hyper-parameters in random forest are either used to increase the predictive power of the model or to make the model faster. I will discuss about the hyper-parameters of `sklearn` built-in random forest functions, since I use this library for all the modeling techniques. The first parameter I have to set to increase the predictive power is the `n_estimators`, this is the number of trees. According to experiences of data scientist in *GitHub* and *DataCamp*, a choice in the range of 10 - 20 is appropriate. In theory, the more trees used, the better the final prediction because the multitude of trees serves to reduce the variance of an already low biased tree ensemble. But the downside of increasing the number of trees is the amount of computing needed. Each prediction of a data point in the test data will have to go through all the individual trees separately. Another important hyper-parameter is `max_features`, which is the maximum number of features Random Forest is allowed to try in an individual tree. On default, this parameter is set as the squared number of the number of features. `min_sample_leaf` determines, like its name already says, the minimum number of leafs that are required to split an internal node. Decreasing this parameter will increase the model complexity and thus lead to a lower bias.

To increase the models speed we can tweak the following parameters;

- `n_jobs`: this parameter tells the engine how many processors it is allowed to use. If it has a value of 1, it can only use one processor. A value of “-1” means that there is no limit.
- `random_state`: makes the model’s output replicable. The model will always produce the same results when it has a definite value of `random_state` and if it has been given the same hyper-parameters and the same training data. I will use the same state for all the models in order to have a valid comparison.
- `oob_score` (also called oob sampling), which is a random forest cross validation method. In this sampling, about one-third of the data is not used to train the model and can be used to evaluate its performance. These samples are called the out of bag samples. It is very similar to the leave-one-out cross-validation method, but almost no additional computational burden goes along with it.

The main limitation of Random Forest is that a large number of trees can make the algorithm to slow and ineffective for real-time predictions. And this can be an issue when implementing an RFC for real-time condition based maintenance. In general, these algorithms are fast to train, but quite slow to create predictions once they are trained. A more accurate prediction requires more trees, which results in a slower model. For the current data set (≈ 300 instances) random forest algorithm is fast enough, but there can certainly be situations where run-time performance is important and other approaches would be preferred. The complete Python script for the RFC model can be found in Appendix C.3.

7.6 Logistic Regression

Logistic regression is another technique borrowed by machine learning from the field of statistics. It is among the commonly used techniques in predictive modeling, and especially for binary classification problems. In this project I am dealing with multi-class output, so in order to use logistic regression as a predictive model I have to create and train a separate classifier for each class and combine the scores of each class to have an overall prediction. Logistic Regression uses sigmoid function. An explanation of logistic regression can begin with an explanation of the standard logistic function. The logistic function is a sigmoid function, which takes any real value between zero and one. It is defined as:

$$\sigma(t) = \frac{e^x}{e^x + 1} = \frac{1}{1 + e^{-x}}$$

Where t is the learned regression function that consists of linear and/or non-linear components of the input data x . Input values are combined using weights or coefficient values (referred to as the Greek letter Beta) to predict the output value y . A key difference from linear regression is that the output value being modeled is a binary values (0 or 1) rather than a numeric value. Below is an example logistic regression equation:

$$y = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

Where y is the predicted output, β_0 is the bias or intercept term and β_1 is the coefficient for the single input value x . Each column in the input data has an associated β coefficient (a constant real value) that must be learned from the training data. This is done using maximum-likelihood estimation. Maximum-likelihood estimation is a common learning algorithm used by a variety of machine learning algorithms, although it does make assumptions about the distribution of the input data. The best coefficients result in a model that predicts a class very close to 1 (e.g. $\text{TSR} > 4000$) for the default class and a value very close to 0 (e.g. $\text{TSR} < 4000$) for the other class. The intuition for maximum-likelihood for logistic regression is that a search procedure seeks values for the coefficients (Beta values) that minimizes the error in the probabilities predicted by the model to those in the data (e.g. probability of 1 if the data is the primary class). Figure 7.7 shows an example of simple sigmoid function with two coefficients.

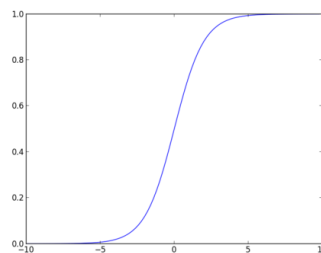


Figure 7.7: Logistic regression function with two coefficients

There are three main parameters to take into account when building the logistic regression model:

- The minimization algorithm used to optimize the best values for the coefficients of the training data can be chosen with the parameter `solver`. All the available algorithms are suitable for our business case except the `liblinear`, because it can not handle polynomial loss and is limited to one-versus-rest schemes. `liblinear` is set as the default solver algorithm in Python, so I manually set it to `lbfgs`. `sag` and `saga` are recommended to use for faster predictions in large data sets. This is only true for features with approximately the same scale. This is the case for a lot of the test data features in our data set, but the size of the data set is not in the order that makes the two latter algorithms more favorable.
- As mentioned earlier, I have set the output value of the data to be multi-class binary variables. Logistic regression models in Python have the ability to not only fit a binary problem for each label, but also to minimize the polynomial loss across the entire probability distribution. The parameter is called `multi_class` and is set to `multinomial`.

The complete Python script of the logistic regression model is included in Appendix C.4. It is worth noting that, in contrast with the previous models, logistic regression has some major disadvantages. It is easy to interpret, doesn't require high computation power and is not hard to implement. But it is not able to handle a large number of categorical features/variables. And as I discussed in Chapter 6, a large amount of features are hot-encoded categorical variables. Also, logistic regression is vulnerable to overfitting and will not perform well with independent variables (e.g. variables that are not correlated with the target variable).

7.7 Model Assessment

At this point, the different types of machine learning algorithms are ranked in Python, using the built-in scikit-learn functions. The key to a fair comparison of machine learning algorithms is ensuring that each algorithm is evaluated in the same way on the same data. For every single model I used 5-fold cross validation, which I explained in section 7.3, and retrieved and summarized the results both numerically and using a box and whisker plot.

The 5-fold cross validation procedure used to evaluate each algorithm, is configured with the same random seed to ensure that the same splits to the training data are performed and that each algorithm is evaluated in precisely the same way. Each algorithm is given a short name, that will be used for the remainder of this report. The code used for the algorithm comparison is as follows:

```
seed = 7
# Prepare cross-validation (cv)
cv = KFold(n_splits = 5, random_state = None)

# Classifiers
models = []
models.append(('LogReg', LogisticRegression()))
models.append(('NN', MLPClassifier())) #Neural Networks
models.append(('RFC', RandomForestClassifier()))
models.append(('XGBoost', GradientBoostingRegressor()))

# iterate over classifiers
models = []
trained_classifiers = []
for name, clf in zip(models, classifiers):
    scores = []
    for train_indices, test_indices in cv.split(X):
```

```

clf.fit(X[train_indices], y[train_indices].ravel())
scores.append( clf.score(X_test, y_test.ravel()) )

min_score = min(scores)
max_score = max(scores)
avg_score = sum(scores) / len(scores)

trained_classifiers.append(clf)
models.append((name, min_score, max_score, avg_score))

```

Model	Min	Max	Average	c-stat
XGBoost	0.66	0.73	0.67	0.69
Random Forest	0.64	0.71	0.66	0.67
Logistic Regression	0.56	0.66	0.57	0.62
Neural Networks	0.63	0.70	0.66	0.65

Table 7.3: Model assesment on: prediction accuracy and the c-statistic

Table 7.3 shows the result of running this code. The min and max column displays the minimum and maximum scores from the cross validation set. The C-statistic (sometimes called the “concordance” statistic or C-index) is a measure of goodness of fit for classification models. It is equal to the area under the Receiver Operating Characteristic (ROC) curve and ranges from 0.5 to 1. We can see that the models have an average performance (.56 – .73). XGBoost and Random Forest clearly score significantly higher than the NN and Logistic Regression estimators. To illustrate this even further I plot the model performance (accuracy) on a whisker plot, see Figure 7.8. The code to calculate the minimum, maximum and average scores for this plot is as follows:

```

for name, model in models:
    kfold = model_selection.KFold(n_splits=10, random_state=seed)
    cv_results = model_selection.cross_val_score(model, X, Y, cv=5)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)

```

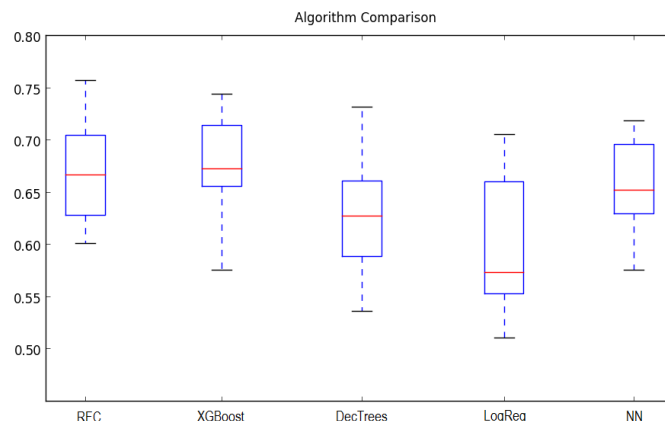


Figure 7.8: Machine learning algorithms comparison

Classifier		Evaluation metric		
		Precision	Recall	F1-score
Random Forest	TSR = 2000	0.63	0.62	0.61
	TSR = 4000	0.50	0.56	0.53
	TSR = 6000	0.61	0.62	0.64
XGBoost	TSR = 2000	0.76	0.71	0.70
	TSR = 4000	0.62	0.58	0.66
	TSR = 6000	0.71	0.72	0.74
Logistic Regression	TSR = 2000	0.61	0.59	0.58
	TSR = 4000	0.54	0.56	0.55
	TSR = 6000	0.62	0.62	0.61

Table 7.4: Evaluation of classifiers for different classes

The assessment of the models gives information on the overall accuracy of the models. To get more insight in the classification performance of the models for the different TSR classes ($TSR = 2000$, $TSR = 4000$, $TSR = 6000$), I calculated the average test scores for each model on the different classes, see Table 7.4. I used the following benchmark on the $f1$ -score (combined metric of precision and recall) to assess the models: 0.5 indicates a very poor model, whereas 0.8 indicates a strong model.

What is striking is that the performance for the class; $TSR = 4000$ is lower than the other two classes. This is due to the fact that the models are less able to make the distinction at the separation line of 4000 flight hours. Random Forest and XGBoost score higher than Logistic Regression. This is because they select important features automatically.

7.7.1 Revised Parameter Settings

In order to improve the created models, I have two options: gather more data and perform feature engineering. Or revising the parameters by performing a hyper-parameter optimization. The first option has usually the greatest pay-off in terms of invested time versus improved performance, but there are no additional work orders I can add to the data set and other data sources are exhausted, I chose to perform hyper-parameter tuning on the existing models. Unlike hyper-parameters, general parameters are trained during the learning process. Hyper-parameters are parameters that are not directly learned within estimators. In scikit-learn they are passed as arguments to the constructor of the estimator classes. In case of the Random Forest, hyper-parameters include the number of decision trees, and the number of features considered by each tree when splitting a node.

Some models allow for specialized, efficient parameter tuning strategies. I will use a strategy called grid search (`GridSearchCV`). It generates candidates from a grid of parameter values that I have to specify with the `param_grid` parameter. Parameter search uses the score function by default to evaluate a parameter setting. Because I am interested in the combined metric of precision and recall, I used the $F1$ -score as input for the grid search. Next I instantiated and fitted the grid search for each model and displayed the best hyper-parameters, and the corresponding performance. For the XGBoost and Random Forest models I used the following `param_grid`.

```
param_grid = {'bootstrap': [True, False],
              'class_weight': ['balanced', None],
              'max_depth': [20, 40, 60, 80, 100, None],
              'max_features': [1, 2, 3, 'log2', 'auto'],
              'min_samples_leaf': [1, 2, 3, 4, 5],
              'min_samples_split': [8, 10, 12, 14, 16],
              'min_weight_fraction_leaf': [0.0, 1.0, 2.0, 3.0, 4.0, 5.0],
```

```

        'n_estimators': [20,50,100,150]
    }

grid_search = GridSearchCV(estimator = rf, param_grid = param_grid,
                           cv = 5, n_jobs = -1, verbose = 2)

grid_search.best_params_

'bootstrap': True,
'class_weight': 'balanced',
'max_depth': 80,
'max_features': 3,
'min_samples_leaf': 5,
'min_samples_split': 12,
'min_weight_fraction_leaf': 0.2,
'n_estimators': 100}
best_grid = grid_search.best_estimator_
grid_accuracy = evaluate(best_grid, test_features, test_labels)

```

For XGBoost I included the additional parameters: `grow_policy` and `scale_pos_weight`. Table 7.5 shows the model performance of the models using the tuned hyper-parameters and the corresponding improvement.

Classifier		Evaluation metric		
		Precision	Recall	F1-score
Random Forest	TSR = 2000	0.69 (+0.06)	0.63 (+0.02)	0.65 (+0.04)
	TSR = 4000	0.50	0.58 (+0.02)	0.53
	TSR = 6000	0.63 (+0.02)	0.65 (+0.03)	0.66 (+0.02)
XGBoost	TSR = 2000	0.83 (+0.07)	0.74 (+0.03)	0.75 (+0.05)
	TSR = 4000	0.63 (+0.02)	0.56 (+0.01)	0.66
	TSR = 6000	0.75 (+0.04)	0.77 (+0.05)	0.77 (+0.03)
Logistic Regression	TSR = 2000	0.62 (+0.01)	0.61 (+0.02)	0.60 (+0.02)
	TSR = 4000	0.56 (+0.02)	0.56	0.56 (+0.01)
	TSR = 6000	0.65 (+0.03)	0.63 (+0.01)	0.61

Table 7.5: Classifier performance after parameter optimization

Intuitively, the accuracy measures the overall performance of the classifier, while precision and recall measure the performance on the positive (resp., negative) class only. Comparing results from the three classifiers I can draw the following conclusions:

- The performance of the XGBoost model is slightly better than those obtained for random forest and logistic regression, both exhibiting quite similar results.
- XGBoost model is able to reach good precision values for both "low TSR" ($p = 0.83$) and "high TSR" ($p = 0.75$) values.
- On the other hand, the recall for the XGBoost model is very low ($r = 0.59$ and $f1 = 0.66$); this means that the probability of false alarm for this classifier is quite high (in $1 - 0.56 = 44\%$ of the cases, the classifier raises a false alarm).

Chapter 8

Results

8.1 Main Findings

In this section I will present the results and main findings of the modeling techniques discussed in the previous chapter. The direct correlation between the input variables and the target variable can best be measured using the bi-variate correlation coefficient. To investigate this, I use the so-called Pearson correlation coefficient. This is the co-variance of the two variables (X and Y) divided by the product of their standard deviations. The coefficient is represented by the Greek letter ρ (rho) and the formula is as follows:

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$$

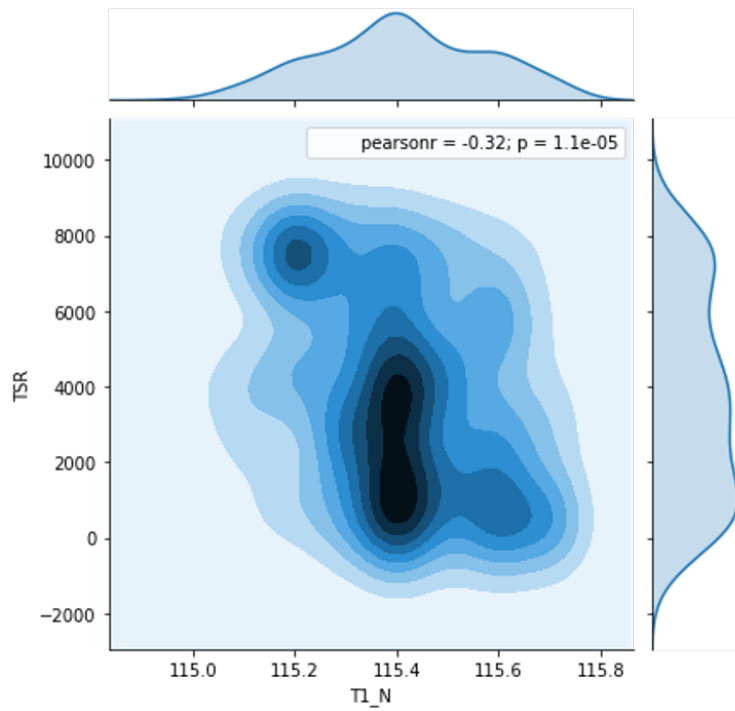
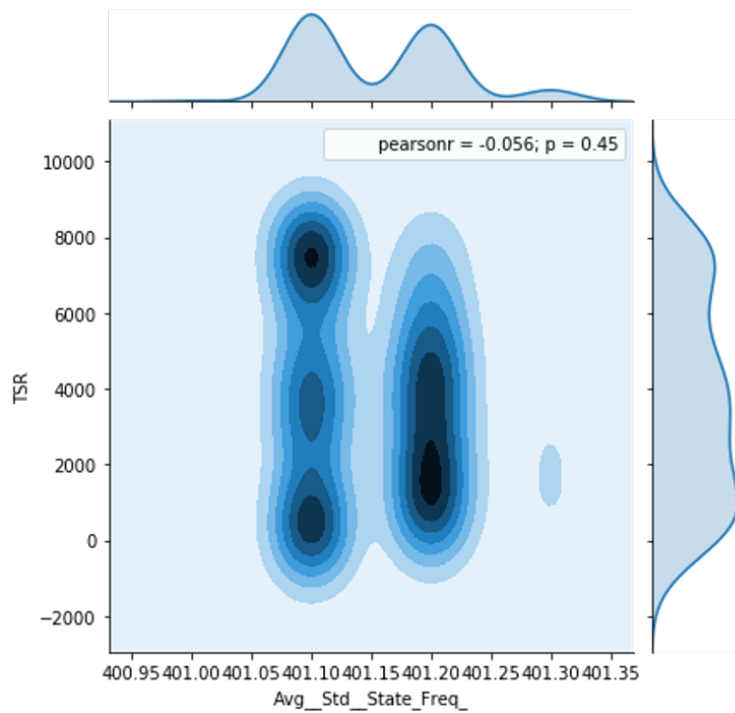
Where cov is the co-variance, and ρ is the standard deviation. Table 8.1 shows the features with the highest correlation coefficient in descending order. Each feature name begins with the name of the test, followed by the recorded measurement in bold. The correlation coefficient ranges from -1 to 1 . A value of 1 implies that a linear relationship between X and Y , with all data points. A value of -1 implies that all data points lie on which Y decreases as X increases. A value of 0 implies that there is no linear correlation between the variables.

The largest linear correlation (-0.317 for $T1_N$) shows that there are no major dependency between the individual features and the final TSR of the component. The data on parts used and amount of hours worked also do not return in the ranking. The **FREQ_REG** appears multiple times in the ranking. Figure 8.1, shows the scatter-plots of the **FREQ_REG** readings against the TSR values.

Feature	Correlation with TSR
FREQ_REG TN 1	-0.317
SHORT_CIRCUIT T2 to T4 Voltage	0.185
PMG_VOLTAGE Exciter Current	-0.185
GEN_LOAD_CHECK I2	-0.179
FREQ_REG Avg Std State Freq	0.175
FREQ_REG Recovery Steady State Freq	-0.17
GEN_PHASE_3 Current	0.15

Table 8.1: Ranking of linear correlation between input features and TSR

Figure 8.1(a) shows a vague S-curve. Most components have a reading of 115.4 on the $TN.1$ test. But components with a higher score tend to have a lower TSR, and components with a lower reading have a slightly higher TSR value. In Figure 8.2(b) it shows that the frequency readings

(a) TN_1 against TSR(b) $Avg_Std_State_Freq$ against TSRFigure 8.1: Seaborn Scatter-plots of `FREQ_REG` features

are clustered around two values: $401.1Hz$ and $401.2Hz$. Although the difference is minimal, the negative Pearson correlation suggest that higher frequencies result in lower TSR. The same phenomenon can be found in other features with frequency related readings.

The input data on employees, parts, and amount of hours worked, no significant individual correlation has been found: .05, .09, and .01 respectively. An interesting finding was the relation between TSR and HDRACC.NO. The account number or code refers to the airline owning the component. Although this variable scores low in the Pearson score [0.02], there is one airline that has a relatively low TSR value. The data set contained 13 components from the account CA003, with a maximum TSR of 4892 hours, see Figure 8.2.

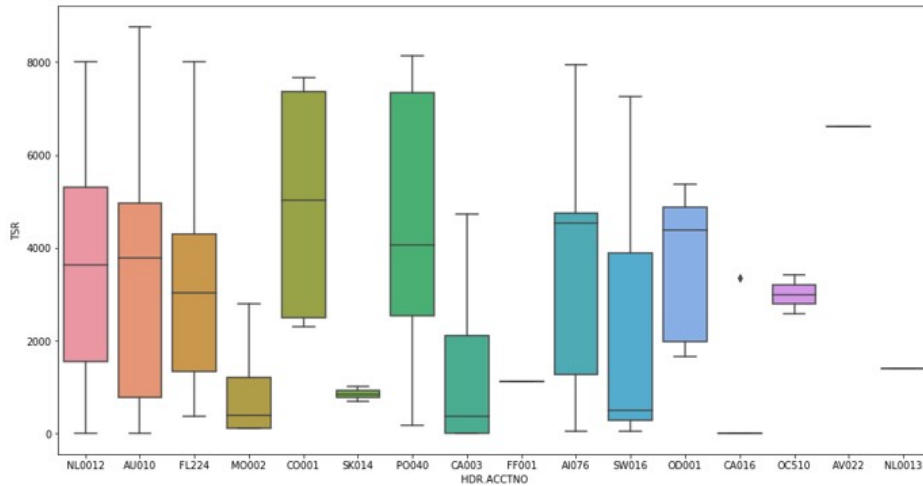


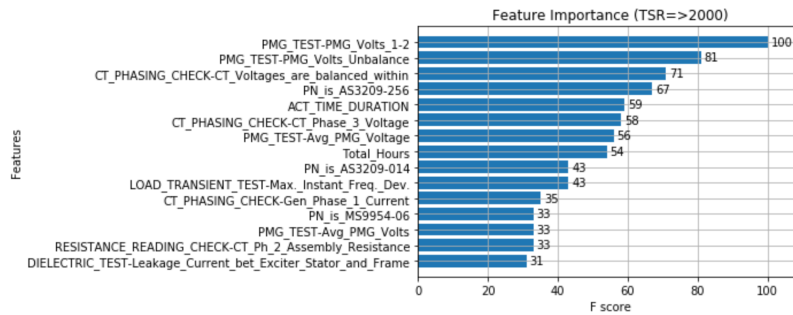
Figure 8.2: Box-plot of the Account Numbers (Airlines)

	Count	Average TSR	Std. TSR
AU010	112	3412	2528
CA003	39	1128	1340
CO001	10	4967	2655
NL0012	31	3529	2402
PO040	30	4494	2672
...			

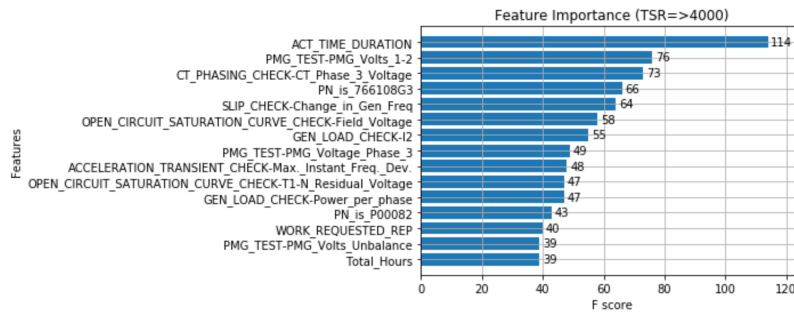
Table 8.2: TSR statistics for different Airlines

8.2 Feature Importance

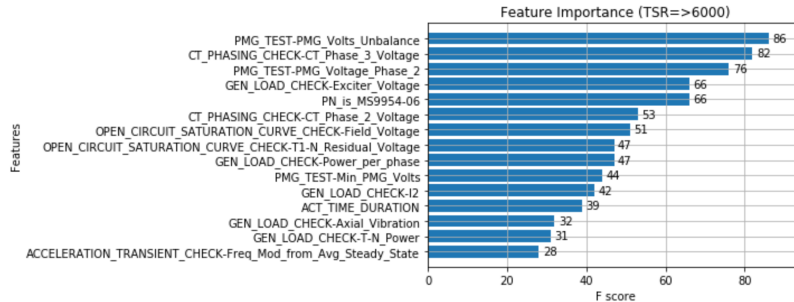
In previous section I reported the results and findings of the linear correlation between input features and the output variable. In this section I will report the more complex relations that underlie the ensemble tree models. These relations are not linear in nature and are best represented by the feature importance of the model. It has been shown that the XGBoost model has the highest prediction accuracy for all TSR classes. The next important step, is to look at the feature importance of this model. Feature importance gives a score for each feature of the data, the higher the score more important or relevant is the feature towards the output variable. Feature importance is an inbuilt class that comes with Tree Based Classifiers, I will be using this class to extract the top 15 features for the XGBoost model.



(a) $TSR = 2000$ class



(b) $TSR = 4000$ class



(c) $TSR = 6000$ class

Figure 8.3: Feature Importance plots for the XGBoost classification model

In Figure 8.3 it shows the ranking of the top 15 features based on the F-score. The F-score is simply calculated by summing the presence of a given feature in all the trees. This is a metric that simply sums up how many times each feature is split on. What emerges from these plots is that a number of features score high in all three classes. Especially from the `PMG_Volts` and the `FREQ_REG` appear high on the importance ranking. This is even more apparent in the display of the XGBoost decision tree. The final decision tree is shown in Figure 8.4. Each box is headed by the name of the feature and the cut-off value for the classification decision. Samples amounts the number of instances at that decision points (from the test set). And the values specify what instances belong to each TSR class. For example `[5, 13, 34, 60]` means that 5 instances are classified to have a TSR of smaller than 2000 flying hours. 13 instances are classified to have a TSR between 2000 and 4000 hours, etc. The final XGBoost trees that classify each class separately are included in Appendix E.

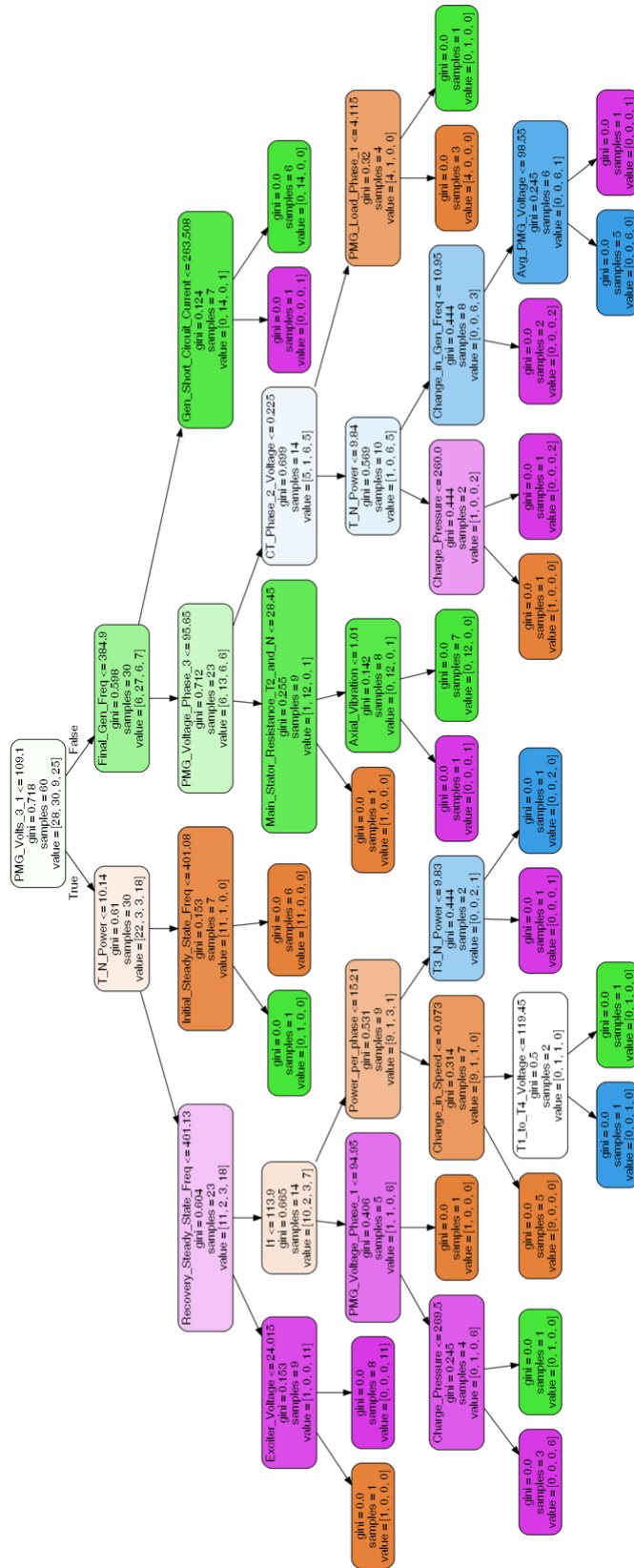


Figure 8.4: Decision tree of the XGBoost model

Chapter 9

Conclusions and Recommendations

In this thesis I have presented a data mining tool for the implementation of condition-based maintenance for the complex component. The tool benefits from the combination of both automatic data analysis techniques and human knowledge, since machine experts are involved in several steps, for example, to select the most important faults for the component at hand or the data relevant to the analysis. The use of this tool to real component test and administrative data allowed me to compare the performance of three different supervised data mining techniques on the prediction of the remaining useful lifetime that is represented by the TSR value. Among the considered classifiers, the XGBoost has been selected as the most effective and promising, thanks to its higher precision on detecting low TSR values (80% precision with a 44% chance of a false alarm) and its interpretability that could help troubleshooting.

However, based on the high chance of false alarm I do not recommend initiating the deployment of the current tool in the maintenance decision making, for a number of reasons. The tool is still in the initial phase of development and is built on available event data from historical work orders. This data can not guarantee an accurate prediction of the health of current and future component work orders, as no trends in the fault mechanisms is taken into account. In addition, a preventive maintenance decision has a major financial and logistical impact. With a high probability of a false alarm, a deployment of the tool will lead to unnecessary maintenance operations and customer contract costs. This can also result in a lack of trust and company-wide adaptation to future, more accurate CBM tools.

For future research there are several areas where additional research can lead to insights to improve results in terms of predictive maintenance, prediction accuracy and cost savings. These areas are as follows

- Fokker wants to implement CBM with prognostic features in the future, however there are some crucial steps that need to be done before this is possible. In general, for a CBM system, a lot of data as well as knowledge about the specific component is needed. So the first step necessary to implement CBM is to decide which data to collect, i.e. what should be considered as input data in the CBM program. In this step it is important to get a thorough understanding of the component in question and decide whether the current collection procedure specified in the CMM manual is sufficient for a remaining lifetime estimation or if new test readings need to be logged. This can be seen in my analysis of the correlation of the features with the component condition, where I could not find any correlations with the TSR. In this case it might be needed to install additional hardware to be able to log other signals that are not available today. The current situation at hand in the analytics department is the availability of a lot of data for analysis but a lack of knowledge

about how to interpret the data. This could be called “data rich – information poor”.

- During my conversations and research at the Schiphol repair shop, it became clear that the use of text and notes is widespread in maintenance documents. These notes often contain important information about the failure mode and the repair tasks that have been taken. Using modern techniques, this information can be converted into useful input data to predicting future failures. Text mining is a well-known technique that can be used to detect patterns and tendencies. Concretely, texts will be automatically structured, dissected, transformed, and subsequently inserted into databases where it can be evaluated and interpreted. In my opinion, investing in text mining can generate a large amount of important data for the application of CBM.
- A cost-benefit analysis on the trade-off between false alarms and false negatives should be conducted. Finding the optimum balance between these two classification errors that results in minimum costs can provide future developers of the tool information to set certain parameters of the model in such a way that this balance is reached. In order to make this analysis, the different costs and prices for making the errors along with their respective probabilities should be investigated.

Since the output of this project will not go through the deployment stage which is the last phase of the CRISP-DM framework, a reflections regarding the process should be made. The main issue is the lack of knowledge regarding the fault causes. Without a physical understanding of the degradation process of a component it is very difficult to analyze data and foresee when failure will occur. Simply relying on deep-learning algorithms to find the underlying causes will not work. Most defects are simply replaced and in general no further investigation of the fault cause is performed. Thus, the root cause of the replacement and failure remains unknown. A better approach would have been to have substantive discussions with maintenance engineers about which data is relevant, and which is not. With this knowledge a number of data streams can then be crossed out and more attention can be focused in collecting data that measures the critical signals. Hereby, reducing the time spent on data preparation and increasing predictive performance.

Chapter 10

References

- Tenning, C. (2011). Designing the 777 Electrical Power System.
- Lawrence, M., Anuj, S., and Gerald, M. K. (1995). Statistical-based or condition-based preventive maintenance? *Journal of Quality in Maintenance Engineering*, Vol. 1 **Issue: 1**, 46-59.
- Rosmaini, A., and Shahrul, K. (2012). An overview of time-based and condition-based maintenance in industrial application. *School of Mechanical Engineering*, Universiti Sains Malaysia.
- Srinivasan, M., and Lee, H. (1996). Production-inventory systems with preventive maintenance. *IIE Trans*, vol. **28**, 879-890.
- McAfee, A., and Brynjolfsson, E. (2012). Big Data: The Management Revolution. *Harvard Business Review*.
- Wang, L., Chu, J., and Mao, W. (2008). A condition-based order-replacement policy for a single-unit system. *Applied Mathematical Modelling*, **2274-2289**.
- Cho, D., and Parlar, M. (1991). A survey of maintenance models for multi-unit systems. *European Journal of Operational Research*, **1-23**.
- Reinertsen, R. (1996). Residual life of technical systems; diagnosis, prediction and life extension. *Reliability Engineering and System Safety*, **23-34**.
- Scarf, P. (1997). On the application of mathematical models in maintenance. *European Journal of Operational Research*, **493-506**.
- Van den Bergh, J., De Bruecker, P., Beliën, J., and Peeters, J. (2013). Aircraft maintenance operations: state of the art. *Faculteit Economie En Bedrijfswetenschappen*.
- Shearer C. (2000). The CRISP-DM model: The new blueprint for data mining.
- Franks, B. (2007). International Institute for Analytics.
- Taghipour, S., and Banjevic, D. (n.d.). Optimal Inspection of a complex system subject to periodic and opportunistic inspections and preventive replacements. *European Journal of Operational Research*, **649-660**.

Jardine., A. K., Lin, D., and Banjevic, D. (n.d.). A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing*, **1483-1510**.

Lin, X., Basten, R. J., Kranenburg, A. A., and Houtum, G. J. (2012). Condition based spare parts supply. *Proceedings of Second International Conference on Mobile Technology*. New York: Applications and Systems.

Geitner, F., and Bloch, H. P. (2012). Machinery Troubleshooting. Machinery Failure Analysis and Troubleshooting. 4th edition. *Oxford: Butterworth-Heinemann*.

Zhu, Q. (2015). Maintenance Optimization for Multi-component Systems under Condition Monitoring. *PhD Thesis*, Eindhoven University of Technology.

Martin, K. F. (1994). A review by discussion of condition monitoring and fault diagnosis in machine tools. *International Journal of Machine Tools and Manufacture*, **527-551**.

Davies, A. (1990). Management Guide to Condition Monitoring in Manufacture. *Institution of Production Engineers* (Great Britain).

Tinga, T. (2010). Application of physical failure models to enable usage an load based maintenance. *Reliability Engineering and System Safety*. 95(10):**1061-1075**.

Michales, K. (2016). MRO Industry outlook. IATA 12th Maintenance Cost Conference *ICF International*

MacDonald, M. (2017) Annual Analyses related to the EU Air Transport Market 2016. **p.109**
 Wyman, O. (2016) MRO Big Data - A lion or a Lamb, Innovation and adoption in aviation MRO, MRO Survey.

Wen, Z. and Liu, Y. (2011). Application of Prognostics and Health Management in Aviation Industry. In *Prognostics and System Health Management Conference (PHM-Shenzhen)*, 2011, **p.1-5**. IEEE

Zhao, C., Chen, Y., Wang, H., and Xiong, H. (2012). Reliability analysis of Aircraft Condition Monitoring Network using an enhanced BDD algorithm. *Chinese Journal of Aeronautics*, 25(6):**925-30**

Byington, C., and Stoelting, P. (2004). A Model-Based Approach to Prognostics and Health Management for Flight Control Actuators. In *IEEE Aerospace Conference*, p.**3551-3562**

Byington, C. S., Watson, M., and Edwards, D. (2004). Data-driven neural network methodology to remaining life predictions for aircraft actuator components. In *IEEE Aerospace Conference Proceedings*, volume 6, p.**3581-3589**

Sun, J., Zuo, H., Wang, W., and Pecht, M. G. (2012). Application of a state space modeling technique to system prognostics based on a health index for condition-based maintenance. *Mechanical Systems and Signal Processing*, 20(5):**1189-1201**

Philips, P., Diston, D., Starr, A., Payne, J., and Pandyra, S. (2010). A review on the optimization of aircraft maintenance with application to landing gears. In *Proceedings of the 11th World Congress on Engineering Asset Management (WCEAM 2009)*, number September, p.**68-76**

Jianhui Luo, J., Pattipani, K., Liu Qiao, L., and Chigusa, S. (2008). Model-Based Prognostics

Techniques Applied to a Suspension System. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 38(5):**1156-1168**

Philips, P., Diston, D. (2011). A knowledge driven approach to aerospace condition monitoring. *Knowledge-Based Systems*, 24(6):**915-927**

Roemer, M., Nwadiogbu, E., and Bloor, G. (2001). Development of diagnostic and prognostic technologies for aerospace health management applications. In *2001 IEEE Aerospace Conference Proceedings (Cat. No.01TH8542)*, volume 6, p.**3139-3147**

Peng, Y., Dong, M., and Zuo, M. J. (2010). Current status of machine prognostics in condition-based maintenance: A review. *International Journal of Advanced Manufacturing*, 50(1-4):**297-313**

Engel, S., Gilmartin, B., Bongort, K., and Hess, A. (2000). Prognostics, the real issues involved with predicting life remaining. In *2000 IEEE Aerospace Conference Proceedings (Cat No.00TH8484)*, volume 6, p.**475-469**

Pecht, M. (2009). Prognostics and health management of electronics. *Encyclopedia of Structural Health Monitoring*, p.**1-13**

Wang, W., and Christer, A. H. (2000). Towards a general condition-based maintenance model for a stochastic dynamic system. *Journal of the Operational Research Society*, 51(2):**145-155**

Heng, A., Zhang, S., Tan, A. C., and Mathew, J. (2009). Rotating machinery prognostics: State of the art, challenges and opportunities. *Mechanical Systems and Signal Processing*, 23(3):**724-739**

Yang, D., Li, H., Hu, Y., Zhao, J., Xiao, H., and Lan, Y. (2016). Vibration condition monitoring system for wind turbine bearings based on noise suppression with multi-point data fusion. *Renewable Energy*, 92:**104-116**

Pham, H. T., and Yang, B. S. (2010). Estimation and forecasting of machine health condition using ARMA/GARCH model. *Mechanical Systems and Signal Processing*, 24(2):**546-558**

Phyu, T. N. (2009). Survey of Classification Techniques in Data Mining. *Proceedings of the International Multi-Conference of Engineers and Computer Scientists 2009 Vol I*.

Thomas, C. Redman (2008). Data Driven: Profiting from Your Most Important Business Asset. *Harvard Business Press, 2008*.

Kuhn, M., Johnson, K. (2013). Applied Predictive Modeling. 1st edition, Springer. p.70

Appendix A

Fokker organigram

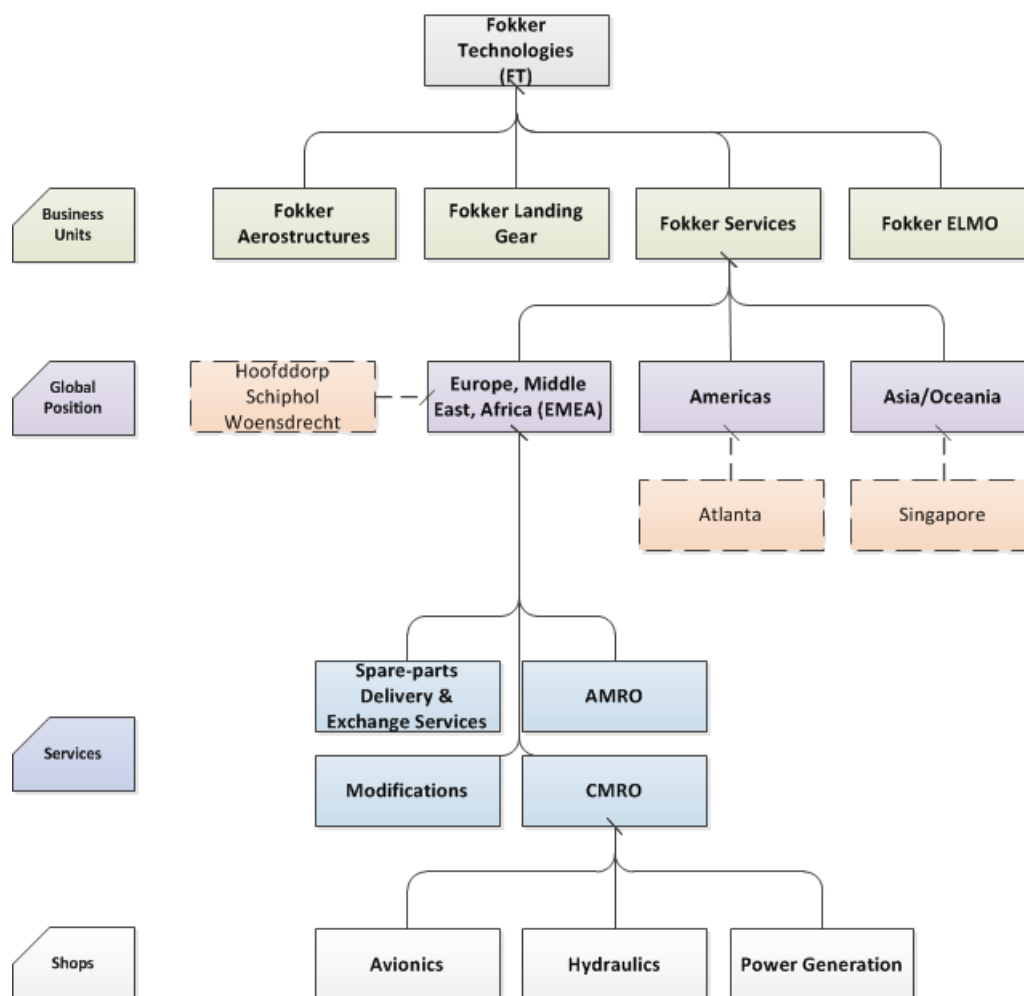


Figure A.1: Organisational chart of Fokker Services and its position in Fokker Technologies

Appendix B

Slip test

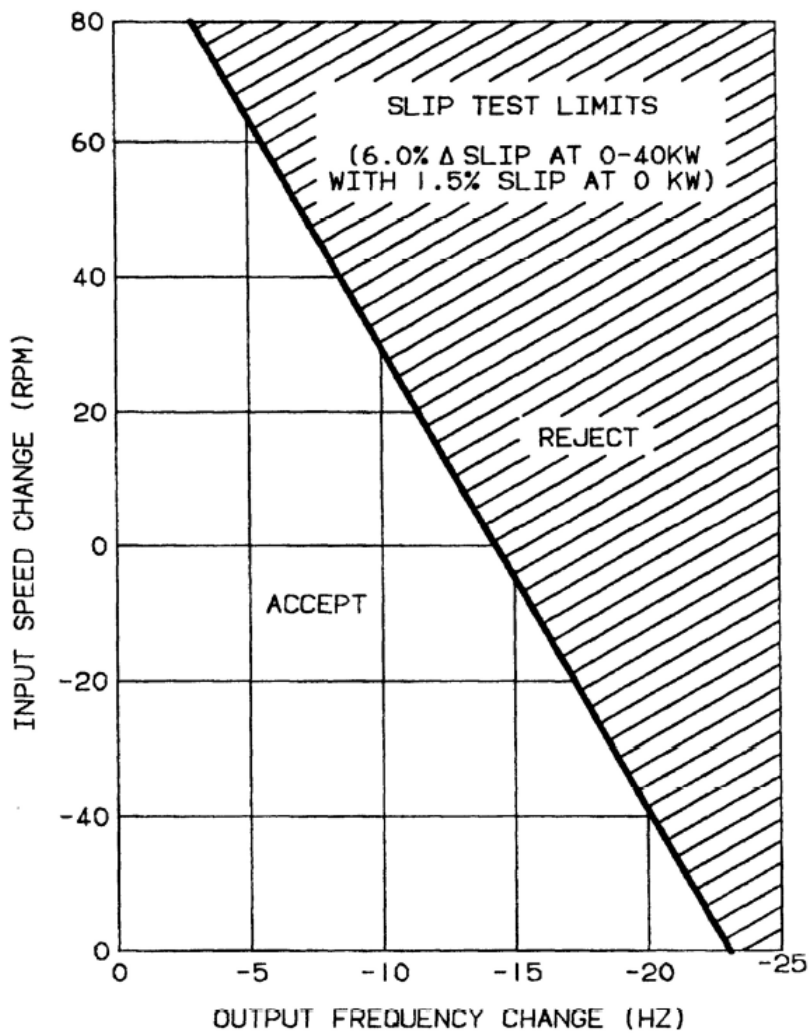


Figure B.1: Slip test Limits

Appendix C

Python Models

C.1 Neural Network

```
In [ ]: # Neural Network Model - NN

import numpy as np
import matplotlib.pyplot as plt

from sklearn import ensemble
from sklearn import datasets
from sklearn.utils import shuffle
from sklearn.metrics import mean_squared_error

#####
# Load data
df = pd.read_csv('IDG_Final_Dataset.csv')
test_size = .44
seed = 4

train, test = train_test_split(data, test_size=test_size, random_state=seed)

Y_train_4000 = train['TSR_4000']
X_train = train[list(data.loc[:, 'Employee_is_17': 'SHORT_CIRCUIT_TEST-Gen_Short_Circuit_Current'])]

Y_test_4000 = test['TSR_4000']
X_test = test[list(data.loc[:, 'Employee_is_17': 'SHORT_CIRCUIT_TEST-Gen_Short_Circuit_Current'])]

#####
# Fit model
clf = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(15,), random_state=1)

clf.fit(X_train, y_train)
MLPClassifier(activation='relu', alpha=1e-05, batch_size='auto',
              beta_1=0.9, beta_2=0.999, early_stopping=False,
              epsilon=1e-08, hidden_layer_sizes=(15,),
              learning_rate='constant', learning_rate_init=0.001,
              max_iter=200, momentum=0.9, n_iter_no_change=10,
              nesterovs_momentum=True, power_t=0.5, random_state=1,
              shuffle=True, solver='lbfgs', tol=0.0001,
              validation_fraction=0.1, verbose=False, warm_start=False)

# compute test set deviance
test_score = np.zeros((params['n_estimators'],), dtype=np.float64)

for i, Y_pred in enumerate(clf.staged_predict(X_test)):
    test_score[i] = clf.loss(Y_test, Y_pred)

#####
# Plot Neural Network
plot_model(clf, to_file='model.png', show_shapes=True, show_layer_names=True)
```

Figure C.1: Neural Network Python code

C.2 XGBoost

```

In [ ]: # Gradient Boost Model - XGBoost

import numpy as np
import matplotlib.pyplot as plt

from sklearn import ensemble
from sklearn import datasets
from sklearn.utils import shuffle
from sklearn.metrics import mean_squared_error

#####
# Load data
df = pd.read_csv('IDG_Final_Dataset.csv')
test_size = .44
seed = 4

train, test = train_test_split(data, test_size=test_size, random_state=seed)

Y_train_4000 = train['TSR_4000']
X_train = train[list(data.loc[:, 'Employee_is_17': 'SHORT_CIRCUIT_TEST-Gen_Short_Circuit_Current'])]

Y_test_4000 = test['TSR_4000']
X_test = test[list(data.loc[:, 'Employee_is_17': 'SHORT_CIRCUIT_TEST-Gen_Short_Circuit_Current'])]

#####
# Fit model
params = {'n_estimators': 500, 'max_depth': 4, 'min_samples_split': 2,
          'learning_rate': 0.01, 'loss': 'ls'}
GradientBoost = ensemble.GradientBoostingRegressor(**params)

GradientBoost.fit(X_train, y_train)
mse = mean_squared_error(Y_test, GradientBoost.predict(X_test))
print("MSE: %.4f" % mse)

#####
# Plot training deviance

# compute test set deviance
test_score = np.zeros((params['n_estimators'],), dtype=np.float64)

for i, Y_pred in enumerate(GradientBoost.staged_predict(X_test)):
    test_score[i] = GradientBoost.loss_(Y_test, Y_pred)

plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.title('Deviance')
plt.plot(np.arange(params['n_estimators']) + 1, GradientBoost.train_score_, 'b-',
         label='Training Set Deviance')
plt.plot(np.arange(params['n_estimators']) + 1, test_score, 'r-',
         label='Test Set Deviance')
plt.legend(loc='upper right')
plt.xlabel('Boosting Iterations')
plt.ylabel('Deviance')

#####
# Plot feature importance
feature_importance = GradientBoost.feature_importances_
# make importances relative to max importance
feature_importance = 100.0 * (feature_importance / feature_importance.max())
sorted_idx = np.argsort(feature_importance)
pos = np.arange(sorted_idx.shape[0]) + .5
plt.subplot(1, 2, 2)
plt.barh(pos, feature_importance[sorted_idx], align='center')
plt.yticks(pos, boston.feature_names[sorted_idx])
plt.xlabel('Relative Importance')
plt.title('Variable Importance')
plt.show()

```

Figure C.2: XGBoost Python code

C.3 Random Forest

```
In [ ]: # Random Forest Classifier - RFC

import numpy as np
import matplotlib.pyplot as plt

from sklearn import ensemble
from sklearn import datasets
from sklearn.utils import shuffle
from sklearn.metrics import mean_squared_error
from sklearn.ensemble import RandomForestClassifier

# #####
# Load data
df = pd.read_csv('IDG_Final_Dataset.csv')
test_size = .44
seed = 4

train, test = train_test_split(data, test_size=test_size, random_state=seed)

Y_train_4000 = train['TSR_4000']
X_train = train[list(data.loc[:, 'Employee_is_17': 'SHORT_CIRCUIT_TEST-Gen_Short_Circuit_Current'])]

Y_test_4000 = test['TSR_4000']
X_test = test[list(data.loc[:, 'Employee_is_17': 'SHORT_CIRCUIT_TEST-Gen_Short_Circuit_Current'])]

# #####
# Fit model
rfc = RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                             max_depth=None, max_features='auto', max_leaf_nodes=None,
                             min_impurity_decrease=0.0, min_impurity_split=None,
                             min_samples_leaf=1, min_samples_split=2,
                             min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=1,
                             oob_score=False, random_state=None, verbose=0,
                             warm_start=False)

rfc.fit(X_train, y_train)
mse = mean_squared_error(Y_test, rfc.predict(X_test))
print("MSE: %.4f" % mse)

# #####
# Plot feature importance
feature_imp = pd.Series(clf.feature_importances_, index=iris.feature_names).sort_values(ascending=False)
feature_imp

%matplotlib inline
# Creating a bar plot
sns.barplot(x=feature_imp, y=feature_imp.index)
# Add labels to your graph
plt.xlabel('Feature Importance Score')
plt.ylabel('Features')
plt.title("Visualizing Important Features")
plt.legend()
plt.show()
```

Figure C.3: Random Forest Classifier Python code

C.4 Logistic Regression

```
In [ ]: # Logistic Regression

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn import preprocessing
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.cross_validation import train_test_split

#####
# Load data
df = pd.read_csv('IDG_Final_Dataset.csv')
test_size = .44
seed = 4

train, test = train_test_split(data, test_size=test_size, random_state=seed)

Y_train_4000 = train['TSR_4000']
X_train = train[list(data.loc[:, 'Employee_is_17': 'SHORT_CIRCUIT_TEST-Gen_Short_Circuit_Current'])]

Y_test_4000 = test['TSR_4000']
X_test = test[list(data.loc[:, 'Employee_is_17': 'SHORT_CIRCUIT_TEST-Gen_Short_Circuit_Current'])]

#####
# Fit model
LogReg = LogisticRegression(random_state=0)
LogReg.fit(X_train, Y_train)

LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                    intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
                    penalty='l2', random_state=0, solver='liblinear', tol=0.0001,
                    verbose=0, warm_start=False)

LogReg.fit(X_train, y_train)
mse = mean_squared_error(Y_test, LogReg.predict(X_test))
print("MSE: %.4f" % mse)

#####
# Plot model performance
print('Accuracy of logistic regression classifier {:.2f}'.format(LogReg.score(X_test, Y_test)))
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

Figure C.4: Logistic Regression Python code

Appendix D

Scikit-learn flowchart: model selection guide

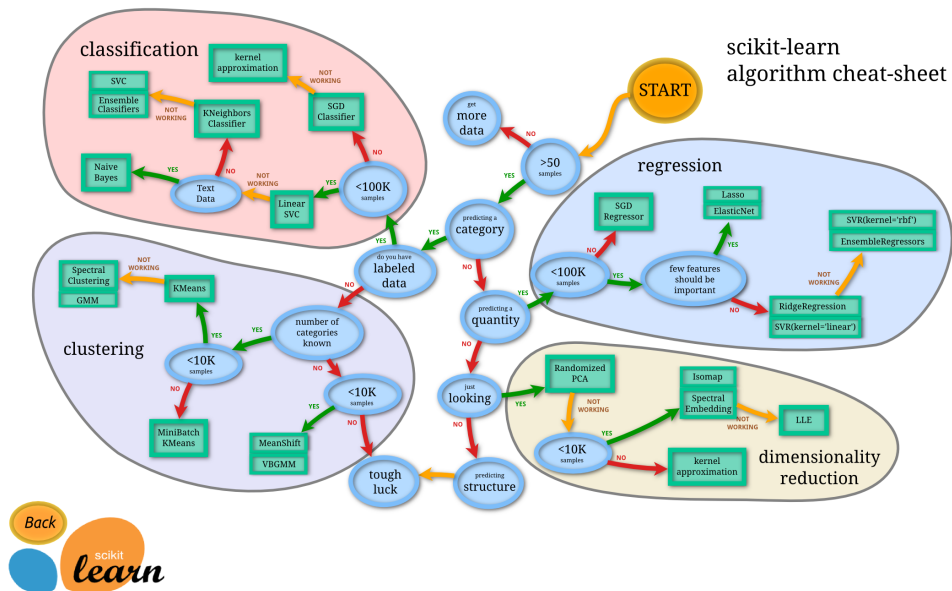


Figure D.1: Model Selection Guide

Appendix E

XGBoost Classification Trees

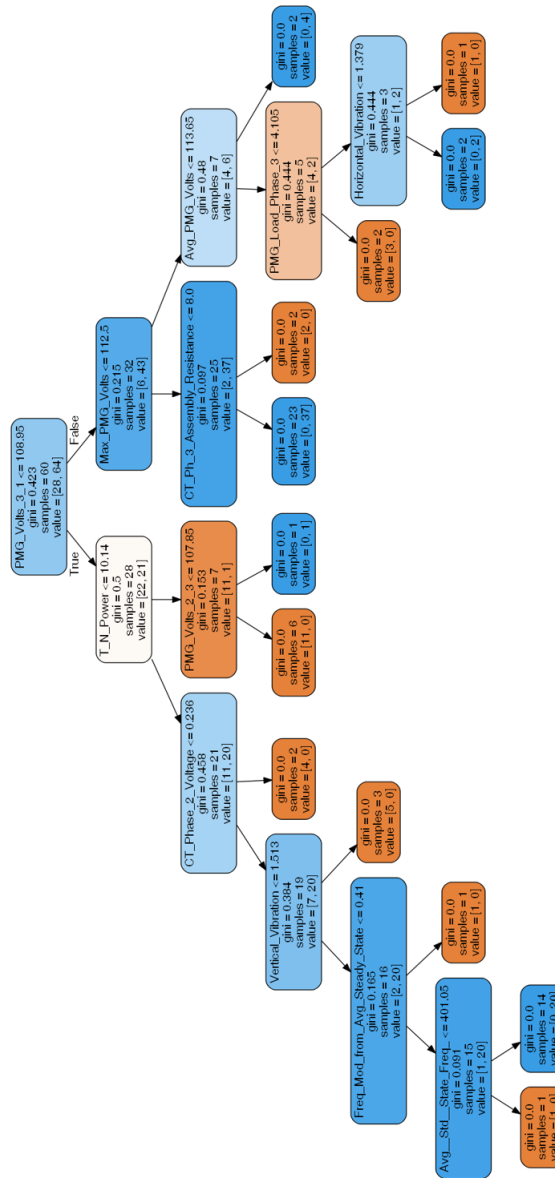


Figure E.1: XGBoost model for TSR = 2000 class

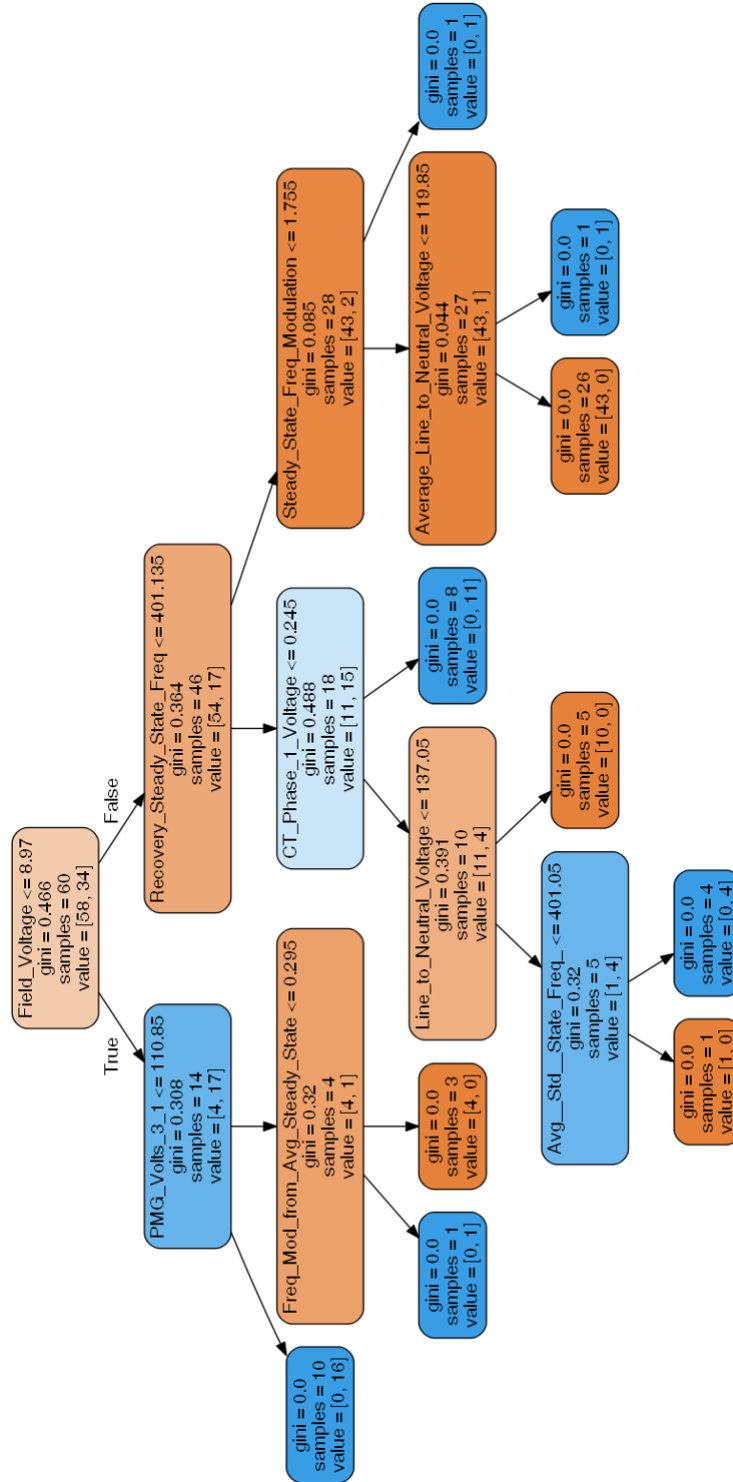


Figure E.2: XGBoost model for TSR = 4000 class

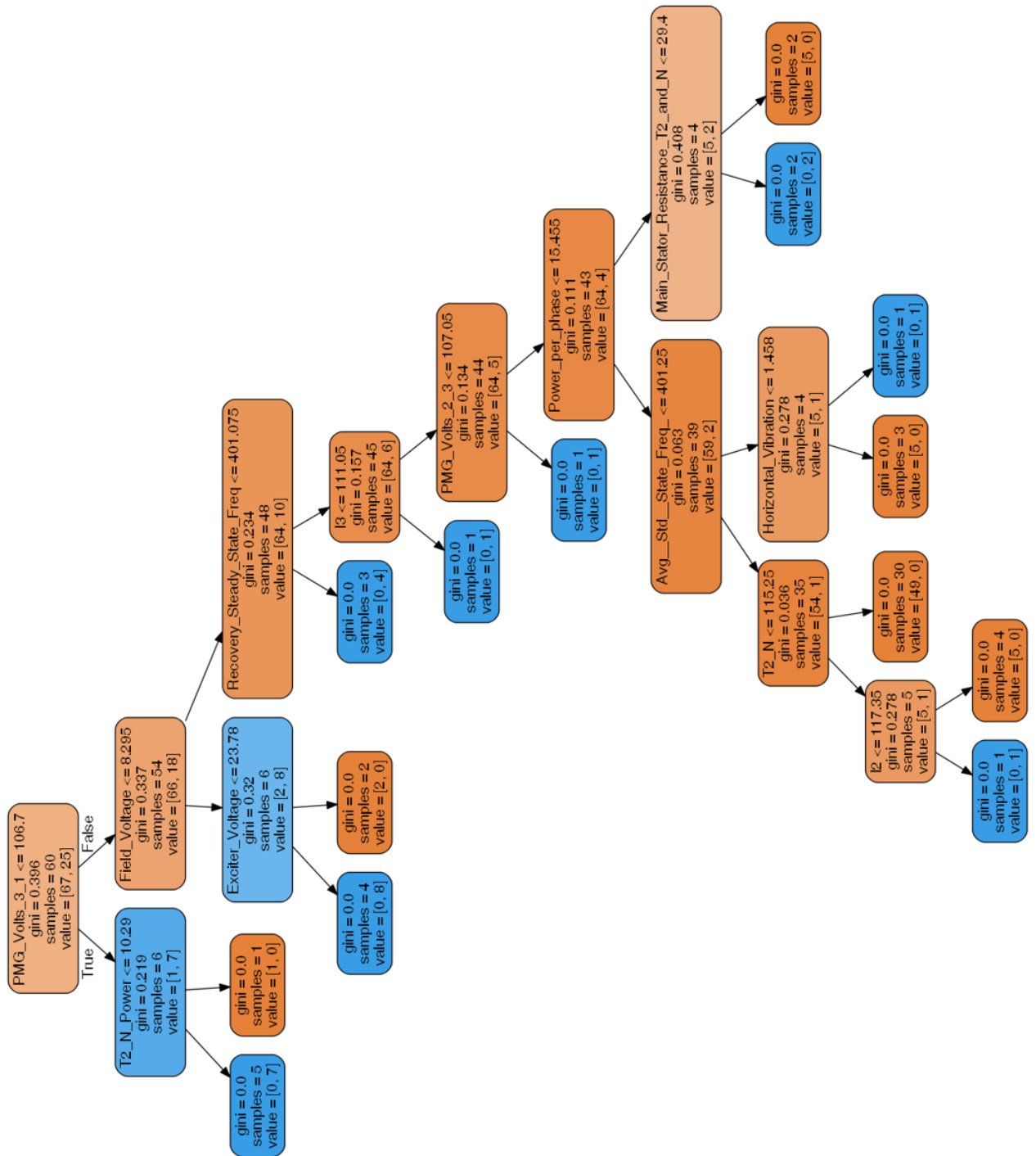


Figure E.3: XGBoost model for TSR = 6000 class