university of groningen / faculty of science and engineering

Predictive maintenance algorithm design for bus door systems

Master's design project

December 8, 2020

Student: Emre Özer

Primary supervisor: Prof. Dr. Ir. Ming Cao

Secondary supervisor: Prof. Dr. Ir. Bayu Jayawardhana

Company supervisor: Kevin Visser

Acknowledgements

The last 14 week period was a fairly exciting and fulfilling experience to me. Being involved in a company, working on my design project for my master 'Mechanical Engineering' at the University of Groningen was a challenging task. I am grateful to all who were sharing this period with me and showing their support.

First of all, I would like to thank my official supervisor, Kevin Visser for providing me with the opportunity to be the part of the Smart Machines project in Ventura Systems.

I would also like to thank my supervisor from the University of Groningen, Professor Ming Cao for his continuous guidance through the project.

Similarly, I would like to express my gratitude to Marten Hoekstra, my "unofficial" supervisor in the project for all of his support. Thank you for teaching me almost everything I have learned during the project and for being in such a satisfying project in this very limited period with me.

I would also like to present my thanks to my colleagues in Ventura Systems for having me as a part of the team; to my team leader, Aebe Meindertsma for introducing me to literally everything about the company, the team and the job and to Gerrit, Watse, Wytze and Germ for sharing this 14 weeks with me.

Last but not least, I would like to thank my parents and my family, for their care, support and wisdom. Even though a distance, thank you for helping me during this entire journey.

Although the project accomplished some important steps, the company still has a long way to go but the potential within the project is vast. I am proud to be a part of the basis of the Smart Machines project in Ventura Systems. I believe the spark we ignited here is a strong foundation to what the company and the industry plan to achieve in the next decade.

Abstract

The decrease of costs for condition monitoring (CM) equipment created a grand opportunity to inspect the state of a system for improved operations and increased efficiency. A major application for CM equipment is error detection and accordingly, error prevention. The enhanced computational power enabled the engineers to estimate overall effective life span for machines and the knowledge of the overall failure times resulted in a new maintenance method, preventive maintenance. The emerging CM technologies promise a more powerful technique called predictive maintenance. Constantly monitoring the system provides exact information about the operations of the machine and enables to foresee any kind of failures. Predictive maintenance is useful for both preventing failures and preventing excessive maintenance.

This study aims to prepare a base algorithm for a predictive maintenance system, planned to be implemented in bus doors. The data collected by Ventura Systems via an endurance test will be analysed to observe the correlations between parameters and to detect patterns. Thereafter based on the initial analysis, a new test set will be designed to gather more useful information. Finally, when the test is done and the data is collected, a predictive model will be generated. Three different prediction models will be created by regression, exponential degradation and neural networks to compare the reliability of all three designs. SensorCloud will be used for data storage and its MathEngine extension will be used for data processing. MATLAB's predictive maintenance and neural network toolboxes will also be used for design and validation.

The study aims to inspect the data obtained from the endurance test, to build a predictive model with that data and to investigate the correlations of the data gathered from different sensors. Performance optimization and external physical effects are out of the scope of the project and will not be included in the study.

Table of Contents

List of Figures

List of Tables

1. Introduction

The project to be conducted is the design of a predictive maintenance algorithm for the electrical and pneumatic bus doors used mostly in public busses. The project is conducted in the company Ventura Systems under the name of Smart Machines Project. The report will include the methods and technologies that are currently being used to build predictive maintenance systems, the problem definition and tests that are done by the company, the approach to the solution of the problem and explanation of the design.

Maintenance is a substantial action in a process to ensure efficiency and safety. Faulty operation of a component in a system may lead to unexpected and non-compensable expenses, as well as creating a safety gap. The system does not have to be a physical or mechanical asset. Processes, actions and even documents such as laws can require maintenance in time. Depending on the application and the type of asset, method of maintenance varies greatly. For instance, in a system that does not cause safety risks for its users and economical lost for its owners, it can be more profitable to wait until the system is inoperable and fix the malfunctioning component while in a system that contains high risk during failures such as aeroplane engines or high-profit loss such as oil pumps, it is desired to ensure the proper operation of the system.

1.1 Background information on maintenance

With the increased cumulative knowledge and the improvements in the technology, it has become possible to build cheaper, larger, faster and smarter maintenance systems. Even though advanced maintenance systems have higher expenses due to its requirements for better equipment, the costs of digital and mechanical equipment are decreasing rapidly. Moore's law suggests that the number of transistors in an integrated circuit doubles every two years. [19] Respectively, the costs of electrical equipment are decreasing constantly. The decrease in costs made condition monitoring (CM) equipment convenient enough to be used widely in the industry. Usage of new tools enabled the development of new techniques for monitoring and maintaining a system.

A mechanical system may need to be fixed due to several reasons such as environmental forcing, operating conditions or deformation of the material. Currently, three main maintenance strategies are being used for this purpose. These are reactive (or run to failure) maintenance (RM), preventive maintenance (PM) and condition-based maintenance (CBM). [18] A basic knowledge of maintenance methods is required to understand the context of the project.

Run-to-failure maintenance
Also known as reactive or corrective maintenance, run-to-failure maintenance method is based on the principle as stated by Keith Mobley: *"If it is not broken, don't fix it"*.[18] It is simply letting the system to operate until it breaks down

or stops working due to a failure. When a failure occurs, the system is repaired and continues to operate. This method has two main disadvantages. The first disadvantage is the lack of safety. Depending on the system, an instant break down may cause a great threat for the personnel nearby. For example, waiting for a ship to sink before repairing is not the best way to maintain the machinery. The second disadvantage is the loss of profit. Unexpected failures may cause great losses to the user. Especially in industries that short times means a large amount of profit such as the oil industry, a sudden failure is intolerably expensive.

Preventive maintenance
To ensure that the maintenance is done before anything breaks, PM can be used as an effective technique. PM depends on setting standard maintenance periods and inspecting the system no matter if there is a problem or not.[9] The method includes tests before usage, planning the failure time for the machine and determining the maintenance period based on that predicted failure time.

The main disadvantage of this method is, it is not possible to predict the effects of different operation conditions on the system.[3] Several tests can be done to foresee the results in different conditions but this still does not guarantee that the system will not fail under extreme conditions. Another disadvantage is the system is advised to go into maintenance if the predicted usage time is fulfilled even if there is no problem on the machine. Excessive maintenance is also an undesired problem since it causes a great loss in efficiency and accordingly, the profit.

Predictive maintenance
As a type of CBM, PdM aims to adapt the economically convenient schedule of corrective maintenance while avoiding failures. [18] The difference from PM is, instead of using general lifetime predictions, PdM uses real-time data of the system to understand the current condition of and to foresee if a failure is about to happen. Since the main principle is based on the monitoring of the component of interest, it is called condition-based maintenance.

The method requires condition-monitoring equipment to analyse the instantaneous situation of the system.[4] The collected data is processed with the PdM algorithm to identify if the system or a part of the system requires special attention. There are several techniques used for predictive maintenance.[23] Those techniques include;

- Vibration monitoring: Used mostly in rotating machines. Very effective technique but not applicable in every system.
- Performance monitoring: Fluctuations in performance gives a strong insight into the condition of the machine.
- Thermal analysis: Used to detect thermal defects in systems.
- Oil analysis: The lubricant can be inspected for microscopic particles to predict failures. This technique is mostly used in bearings.

- Acoustic emission: The change in the noise of mechanical systems can be used to gain information about the condition.
- Corrosion monitoring: Ultrasonic measurements are used to investigate systems under a corrosive environment such as pipelines.[23]

The comparison of maintenance methods
A general comparison of the three mentioned maintenance methods can be seen in table 1.1. The desired maintenance method can be selected by the maintainer according to those advantages and disadvantages.

	RM	**PM**	**PdM**
Pros	-Cheap and simple -Basic technology	-Reliable -Low overtime costs -Reduces waste/energy -Low unscheduled downtime	-Low unscheduled downtime -Real-time information -Optimum maintenance time -Very high reliability
Cons	-Unscheduled downtime -Overtime labour -Unsafe system	-Excessive maintenance -More complex than RM -Long implementation time	-Requires high technology -High costs -Requires high knowledge and skill -Long implementation time

Table 1.1: The comparison of RM, PM and PdM.[3]

1.2 Overview on the maintenance market

Despite the fact that more efficient maintenance tools are being developed everyday, reactive maintenance still occupies the largest share in the industry. According to United States Department of Energy, the aforementioned maintenance methods have a share in the market for 55%, 31% and 12% for reactive, preventive and predictive maintenance, respectively. [27] However, with the relatively small percentage in the market, PdM still holds an annual market size of 3.18 billion dollars by 2018 and it is expected to reach 28.24 billion by the end of 2025.[1]

It can be seen that due to the expenses of implementing an advanced maintenance program, basic methods are still considered as adequate in the majority of the market albeit the opportunity for maintenance cost savings. PM can reduce the maintenance expenses by approximately 12% to 18% relative to RM. However, it does not mean that PM can be used solely in a system. In spite it aims to prevent failures before they occur, random failures or infant mortalities can still occur, causing a corrective action. The occurrance of random failures can be limited by decreasing the maintenance intervals, yet this may cause greater expense.[22] The cost - maintenance diagram for PM is given in Figure 1.1. As the frequency decreases, more RM actions are needed which will increase the cost due to failures. As frequency increases, more excessive maintenance operations will be done which will increase the cost due to increased

downtime. Optimum point is where the combination of PM and RM expenses are minimum.

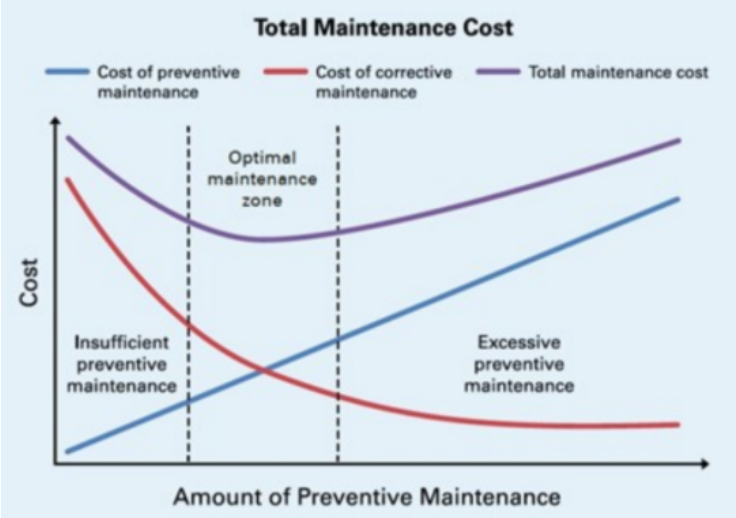


Figure 1.1: Optimum maintenance frequency for PM. (TUV Rheinland, 2019)[22]

PdM promises an even higher cost saving. Relative to PM, it can save approximately 8% - 12% in costs, meaning that 25% - 30% compared to RM. In addition, it can eliminate breakdowns for up to 75%, reduce downtime for up to 45% and increase production for 25%, promising 10 times of return on investment. [27]

The costs for PdM are still high due to the expenses for CM equipment and implementation costs. As the market grows, the equipment costs will decrease greatly and implementation procedure can be standardized. It is obvious that by the end of 2025, PdM will have a much larger share in the market.

1.3 The Smart Machines project

Smart Machines project is a mutual project of the companies of Innovation Cluster Drachten. The ultimate goal is to understand how the system of interest performs under various conditions and to be able to make predictions about the state of it.
This report focuses on the initial design step of the Smart Machines project that will be done within the company Ventura Systems. The company is a bus door and door mechanism provider. The 'Smart Machines' project was launched in 2019 and the plan is to fully integrate the predictive maintenance system on their products by the end of 2021. The company intends to reduce the extra costs by reducing the downtime and unexpected failures of the product and to enhance the product quality by predicting the potential failures. This report covers the 14 weeks of the project starting from September 2020.

1.3.1 The company

Ventura Systems is a company that is specialized in the design and manufacturing of doors for bus, tram and metro. The company currently designs inward swinging (ISD), outward swinging (OSD), plug sliding (PSD) and rapid sliding doors (RSD).[28]

The company is a member of the Innovation Cluster Drachten since 2017 and it is currently working on the design of a predictive maintenance system for their bus doors.

1.3.2 Plan of the company for the 'Smart Machines' project

The companies plan is to conduct a long term study on the subject. This report will only focus on a part in the project that will last approximately 4 months.

The company divided the project into two parts. This report will include the first part of the project. This part includes the "extensive monitoring" tests meaning that;

- Monitoring an endurance test in the company
- Determining the important parameters and the sensors required to measure those parameters.
- Local storage of the sensor data
- Analysis of the data
- Design of a preliminary predictive maintenance algorithm

The second part, which will start after the first part ends includes "condition monitoring". Condition monitoring will include;

- Monitoring on real busses around the world
- Determining the influential parameters in real systems
- Obtaining privacy, safety and location information where no network is available.
- Design and implementation of the predictive maintenance system

Aim of the company with the "Smart Machines" project

The project expects to obtain valuable data to provide improved experience on the users. The doors that are manufactured in the company are being used all over the world, in very different conditions including extreme hot(durable up to 70^oC) and cold(durable up to -30^oC). The plan is to gain more insight into the behaviour of the doors in different conditions to predict and prevent failures.

The second goal of the project is the optimization of the door systems for better operations. Learning the behaviour of a door under different circumstances can provide substantial information for product development.

The third goal is to increase the uptime for doors. Understanding door behaviour means understanding the reasons for failures, leading to prevent unexpected downtimes. This would decrease the cost for the customer by preventing excessive maintenance and increase income for the company.

Another goal is to collect real-time performance data worldwide from bus doors, providing more information and also letting the company know exactly what is happening at any moment.

Finally, the last goal is to adapt to the changing market. In an autonomous vehicle, a driver will not be there to handle dangerous situations. A smart door is necessary for this kind of system to ensure user safety.

In conclusion, the main goal of the project is to extend the life cycle of the product and gain more information to improve the product and processes.

1.4 The scope and the goals of the design project

The purpose of the project is to design an algorithm based on the available data that decides when the door system requires maintenance. The optimization of this process includes the elimination of unnecessary(excessive) maintenance periods of the doors.

The initial data was gathered by the company by an endurance test that was ended in June 2020. The data will be used to understand the behaviour of the doors during operation so that it would be possible to predict when a failure is possible.

The main motivation is to create a smart product that can adapt to severely varying conditions such as temperature, force, vibrations etc. and increase the life cycle and uptime for the bus doors since a failure in the door means that the vehicle itself is unusable until the problem is fixed. Collecting more extensive data through this project is the key to achieve this purpose.

A clear statement of the expected goal can be made after clarifying the concepts about maintenance and defining the context of the project. The planned outcome of the project can be described as;

"Designing a predictive maintenance algorithm for bus door systems."

In order to achieve this goal, the study will be divided into three consecutive steps. Those steps can be explained briefly as;

1. **Preliminary design and research**
   The question that needs to be answered in this step is, *"what are the required parameters to build an accurate and reliable predictive model?"* The initial step is to investigate the available data within the company and prepare a design to determine the deficiencies and to have a deeper understanding of the door mechanism. Most importantly, a clear literature

review including case studies and guidelines from software suppliers is conducted to collect information about the progression of a PdM project. The results are expected to provide an answer to the question.

2. **Design of the endurance test and planning the analysis**
   After determining the necessary information, the next step is to determine *"how to obtain and process the required information?"*
   Experimental design based on the required sensor information is explained in Section 3. The algorithms used in the project are also explained in the section.

3. **Design and validation of the predictive model**
   Ultimately, the goal is to design a PdM algorithm with high accuracy and efficiency. To achieve this final goal, the question that needs to be answered is *"how accurate is the algorithm?"*
   Comparison of the design proposals and results are evaluated in Section 4 to determine the most efficient solution to the given problem.

The three steps that were planned to achieve the ultimate goal contain several questions within them. Those questions can be considered as substeps of the project. Searching for answers to those questions enables a more clear vision of the planning. The sub-questions of the preliminary design and analysis step are;

- *When does a mechanism in the door system fail?*
  An electrical door is a complex system, which is a combination of several smaller systems. Any of those systems can have failures in time. Inspecting the behaviour of the door on its way to failure gives valuable insight about the life of a door.

- *What indicates a failure?*
  This question seeks the information of an indication in the data, which warns the user before a potential failure. Such a parameter can easily be used for a predictive algorithm.

- *What kind of information is required to make accurate predictions?*
  Besides the data from the door, the lifetime of a component is influenced strongly by environmental factors. The question inspects the influential parameters that determine the life of a door.

After acquiring adequate information from the first step by answering those three questions, the methodology can be planned. Similarly, the study plans to find answers to several questions in this step.

- *How can we measure the necessary data in a test?*
  After determining important parameters, it is also a tough challenge to design an experiment that provides the information. Therefore this question addresses a vital problem in the project.

- *What kind of tools can be used for analysis?*
  After acquiring the data, the next problem is what to do with it. Several methods are investigated in Section 2 of this report to find a suitable analysis tool and algorithm for this purpose.

The final step aims to solve the problems mentioned at this point. In that sense the design step of the project is based on the questions;

- *Which feature(s) provide information for an efficient model?*
  Feature selection for a predictive algorithm. The goal is to identify the parameters that provide a clear indication of the system.
- *How to combine the parameters or use them separately for failure predictions?*
  Since the door contains many systems that can potentially fail, it is important to understand the indications in the data that point out different types of failures. Also, understanding the interaction of the parameters is necessary for improved prediction accuracy.
- *What is the correlation between parameters?*
  The relation between measured properties from sensors such as stress or acceleration indicates an opportunity to increase the understanding of the system. This question is also an important step for future works.
- *What kind of algorithm gives the most accurate results for this system?*
  Among several design proposals, the most efficient algorithm must be selected for real applications.

The design steps that lead to the ultimate goal, designing a PdM system, are planned to be achieved by answering these questions. As aforementioned, the report will be investigating the first part of the "Smart Machines" project. Within that section, the main goal is to provide a base design and a road map to the company for the construction of the fully operable PdM system, while capturing valuable information from the endurance tests.

The initial goal in this context is the identification of the parameters. To decide what kind of sensors will be used to observe, to measure and to analyse the data, it is essential to have an insight on the impact of the parameters on the life of the door system. This information will be used in the design of test systems in Ventura and in the real busses in the future.

The second goal is to build and to examine a PdM algorithm based on the limited data. If this goal is achieved successfully, the algorithm can easily be improved for a more comprehensive model. Since the ultimate goal for the company is the improve the efficiency of maintenance periods, this step carries vital importance.

Eventually, selecting the most efficient option among the PdM models will be the final step. It is not possible to examine all the alternative methods. Yet, having overall information about the several main types of prediction algorithms can be a piece of valuable information.

1.5 Overview of the structure of the report

The first chapter introduces the definition and a brief history of predictive maintenance and providing general information on maintenance methods. The differences between maintenance methods, disadvantages and advantages are also

being investigated in the chapter. The current system of the company, their goal with the project and their current progress on the project is also briefly explained. One last thing that is mentioned in the first chapter is the scope and aim of this project and how it will be implemented in the companies long term 'Smart Machines' project.

The second chapter demonstrates the relevant literature, current technologies and methods used to build an efficient maintenance algorithm. Similar works, projects, guides, standards and studies are explained in this chapter to provide an insight for the upcoming project. Programming and design tools are also introduced in this chapter with the explanation of the approach to the design problem.

In the third chapter, the road map of the project is explained in details. The methods used to build the initial design is given and its results are investigated for the next step: experimental design. The conclusions made for the design of the test and the structure of the final design proposal based on the test is given in this chapter.

The fourth chapter is the explanation of the results of the designed predictive model and the performance comparison of the proposed alternatives, while explaining the correlations and relations between the statistical features of the data acquired from the sensors to determine if it is possible to reduce the number of sensors. The discussion based on the performance analysis and estimation of costs are included in the chapter.

The fifth chapter contains the conclusions obtained from the analysis of the data, recommendations to the company and the possibilities of future research on the topic.

2. Review of literature

Within the 4th industrial revolution, smart systems gained importance due to their ability to greatly increase the performance. Reduced device costs and increased computation capability enabled the usage of smart systems and in a very short time, many studies related to Industry 4.0 have been conducted.
One of the most useful results of the smart industry is predictive maintenance and several companies, institutions and individuals have been trying to understand the concept. Theoretical explanations, base concepts of maintenance, case studies and proposed solutions and the tools that are being used for this purpose are investigated in this section.

2.1 Maintenance methods and PdM models

A clear explanation of the issue is given by MATLAB in the documents and guidelines written for PdM applications. Those documents include guides and suggestions for a PdM project.

The maintenance strategies which were aforementioned in Section 1.1 were defined in MATLAB's introductory document[14] to PdM as;
Reactive maintenance: Waiting for a failure to repair a machine. Generally used in inexpensive systems but not suitable for complex designs due to safety and costs.
Preventive maintenance: Most commonly used maintenance method in large organizations. Inefficient due to early maintenance periods.
Predictive maintenance: Optimum method for planning maintenance operations and identifying faulty parts.

Comparison of these three methods can be seen in Figure 2.1, similar to Table 1.1.

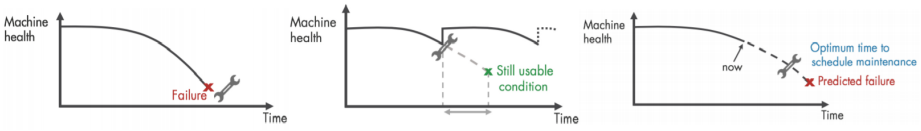


Figure 2.1: Comparison of three maintenance strategies: run to failure, preventive and predictive. (MATLAB, 2019)[14]

In addition to definitions, a general plan of approach is suggested by MATLAB. According to their documentation, PdM is a recursive process rather than a straight line. The model needs to be updated continuously during the training process. The updated model may require new data and new training. A representation of the workflow of a PdM project, retrieved from a white paper of MATLAB is given in Figure 2.2.[13]

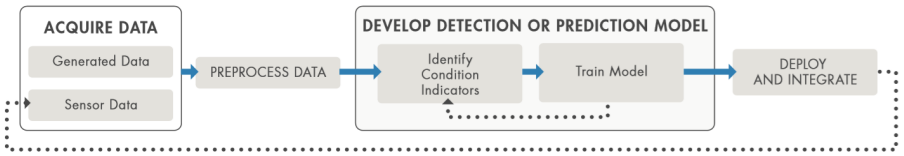


Figure 2.2: Workflow of a PdM project. (MATLAB, 2018)[13]

The initial step is to access sensor data, obtained from tests or field measurements. Those data may include time-domain, frequency-domain or time-frequency-domain features. Mean, standard deviation, kurtosis, and skewness of the data are some of the time-domain features that can be measured. Frequency-domain data includes vibration frequencies, peak-to-peak values, harmonics. Frequency-domain data that changes over time can be captured by time-frequency domain features. The change of vibration frequency as the machine starts to break down is an example of this kind of data.[16]

The collected data needs to be pre-processed afterwards. Before investigating the features of interest, the data needs to be cleared of outliers and noise.

The main purpose of investigating those features is, to be able to identify different types of failures when the machine is running in a certain condition. If a system starts to show a specific type of behaviour as it tends to break down, it is possible to estimate the RUL by comparing the current condition and the failure condition. It also enables to estimate the type of error that can occur so that the problematic parts can be fixed.[16]

The last step is to train the model with the data. After identifying different failure models and the condition of the machine when it is close to the failure points, it is possible to estimate the RUL of a machine by using its current data.[13]

After identifying the steps, a clear definition of algorithms and models can be given. Besides categorizing maintenance algorithms, PdM itself can be divided into several categories based on the availability of the data and the desired method of the prediction algorithm. According to MATLAB, methods to estimate the RUL can be divided into three main categories. Similarity model, survival model and degradation model.[15]

Similarity model is available when a complete test history of a machine is available. Data set from the initial operation of a machine until the breaking down point makes it available to generate an estimation model which assumes the machine will degrade similarly to test data. It is possible to determine when the machine will need maintenance by comparing the current history of the machine with pre-recorded histories of tests. Figure 2.3 graphically shows the comparison of current data with prior information.

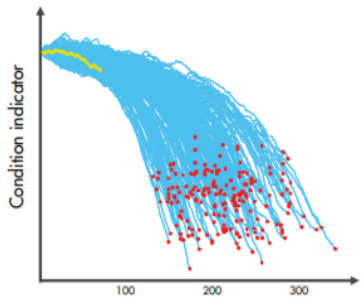


Figure 2.3: Similarity model for RUL estimation. (MATLAB, 2019)[15]

The yellow line is the current situation of the machine. Blue lines are run-to-failure histories of tests and red dots are breaking points. It is possible to predict the degradation trajectory and the time to break down.

Similarity model is divided into three subcategories, hash, pairwise and residual similarity models. These are methods to connect the data history to the prediction model. Hash similarity model uses the entire data for prediction by using their fixes properties such as mean, standard deviation or maximum and minimum values. Pairwise similarity models use several, most correlated trajectories from the data history to predict the RUL of the machine. Residual similarity model uses ARMA or ARIMA algorithms to fit the data for prediction.[15]

Degradation model is used when there is data for a period of a healthy state. The trajectory of the condition of the machine can be used to generate a linear or exponential degradation prediction. Figure 2.4 represents an example degradation prediction for an aircraft engine.[15]

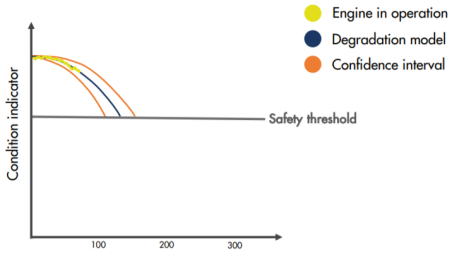


Figure 2.4: Degradation model for RUL estimation. (MATLAB, 2019)[15]

According to the current situation of the engine(yellow line), an exponential prediction is generated(blue line) with a confidence interval(orange lines). A safety threshold is used to determine when the maintenance is required.

Survival model is based on solely failure data. The current condition of the machine is compared to the condition data at failure from tests. When the compared data sets are similar enough, it indicates that the machine is close to failure point and maintenance is required.[15]

2.2 Failure modes

To understand the necessity of maintenance on a system, the causes of failures must be well understood. Nowlan et al. studied the reasons for failure and classified the failure types. According to this study, 89% of the failures occur randomly while only 11% failures are related to ageing. [21]
Both random and age-related failures have 3 models depending on their failure behaviour. Age-related failure models are listed as **Bathtub**, **Wear out** and **Fatigue** while random failures are **Break in period**, **Random** and **Infant mortality**. The failure patterns of those models depending on time is given in Figure 2.5. [11]
Those patterns can be briefly explained as;
Age related

Bathtub Model: High probability of early faults, low-risk region and a wear-out region after ageing. 4% of the total failures.

Wear out: Slowly increasing failure probability followed by a wear-out region. 2% of the total failures.

Fatigue: Steady increase in failure probability. 5% of the total failures.

Random

Break-in period: Low failure rate at the start, steep increase to a steady rate after a short time. 7% of the total failures.

Random: Constant failure rate from the beginning to the end. 14% of the total failures.

Infant mortality: High risk of failures at the beginning of the operation, decreasing to a lower steady level after time. Usually seen in electrical systems. 68% of the total failures.

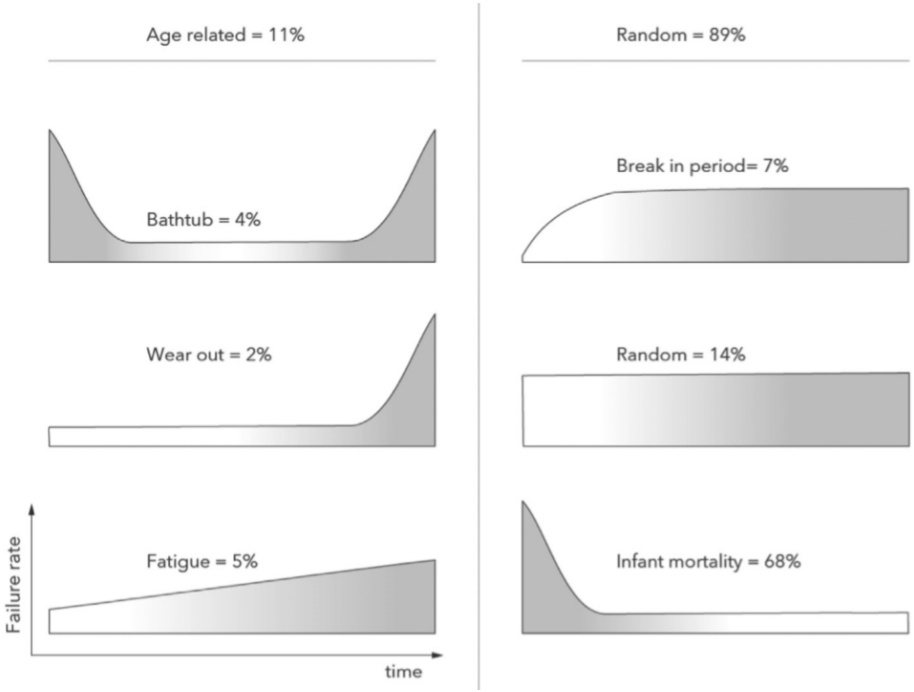


Figure 2.5: Failure rate patterns. (Jimenez et al., 2020)[11][21]

2.3 Successful implementations and case studies

There have been several academic studies focused on the complete implementation of a PdM program. For instance, Jimenez et al. published a case study where a predictive maintenance algorithm was created for a seismic survey ship.[11] The study was based on three types of data: lube oil, vibration and performance. The lube oil data included 624 readings from the test between 2013 and 2018. Several parameters such as additives, viscosity and ratings were selected as control parameters through time. Vibration data was obtained by a condition monitoring system. These measurements were used to detect the problems in rotating parts of the system. Changes of the behaviours of pumps, shafts, engines, fans and compressors were examined through time using vibration analysis. Performance data is the main type of data that was used to build a PdM algorithm.

After the collection phase, the changes in oil, vibration and performance data were analysed to determine a threshold for maintenance. The study also included the development of an AI that suggest the corrective action when a failure is possible.

Daniyan et al. also conducted a similar study for railcars.[10] Wheel acceleration data was used as the primary source of data in the study. The acceleration(and the vibration) data was then used to diagnose the faults and estimate the remaining useful life(RUL) of the wheels. The expected numbers of failures within the operational cycles were then documented. The predictions made in

this study, which gives the estimated RUL for inner and outer bearings can be seen in Figure 2.6.

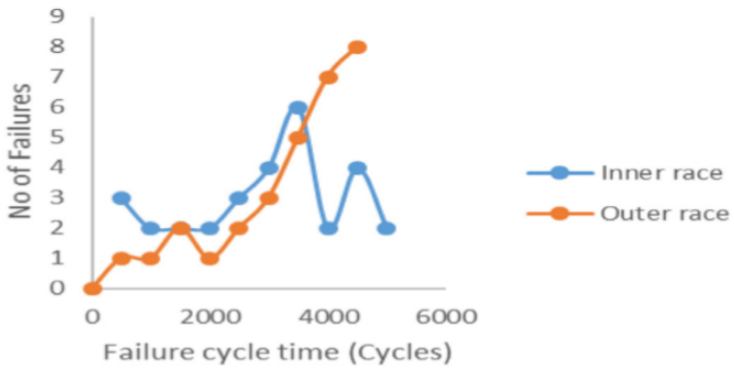


Figure 2.6: Expected failure numbers for inner and outer races of the bearings. (Daniyan et al., 2020)[10]

Following the diagnostic tools for bearings, in another case study by Daniyan et al. explained the usage of artificial intelligence in learning factories for CBM. The learning algorithm is based on the Levenberg Marquardt algorithm, used in foreseeing the change in temperatures and predicting the RUL based on the temperature data.[6] The comparison graphs of the predicted outcomes and real outcomes can be seen in Figure 2.7.

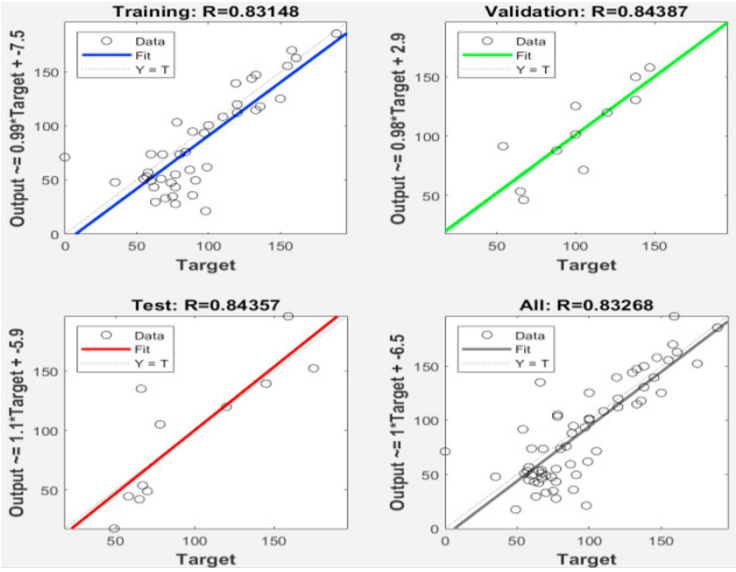


Figure 2.7: Correlations of estimated and real outputs of the AI-based PdM model. (Daniyan et al., 2020)[6]

The figures represent the results in the training set, validation set, test set and overall progress. A correlation number (R) close to 1 means better prediction. The paper suggests that with larger data, it is possible to improve the

correlation accuracy.[6]

2.4 Algorithms and performances

PdM is an application of machine learning and the algorithms used in a model can be altered. Depending on the application, different learning methods may provide better performance. The computational power, data availability and performance criteria are the decision parameters for an algorithm.

It is possible to use simple methods to build a PdM algorithm. Whitaker et al. demonstrate an application of PCA and LDA for predicting the tool life. [29] On a simple example based on the power measurements, the study demonstrated the classification of faulty and healthy operation conditions based on the PCA performed on the measured data. The analysis enabled the classification of failures with 100% accuracy. [29] It is possible to determine if the tools need maintenance by only adding a threshold on the trajectory of the principal component. As the health indicator exceeds the threshold, the tools need to be replaced. The algorithm is relatively simple but effective. Even though this study was conducted on a simple application, basic methods can be effective in more complex designs too. Nevertheless, there are complex methods to generate an algorithm that works with better precision.

As an example, Wu et al. suggested the use of neural networks to create a prediction algorithm.[30] The recurrent neural network algorithm is modified with LSTM(long short term memory) cells for computational convenience. A case study on bearings have been done and the resulting confusion matrix of the test(30%) and training(70%) sets are given in Figure 2.8.

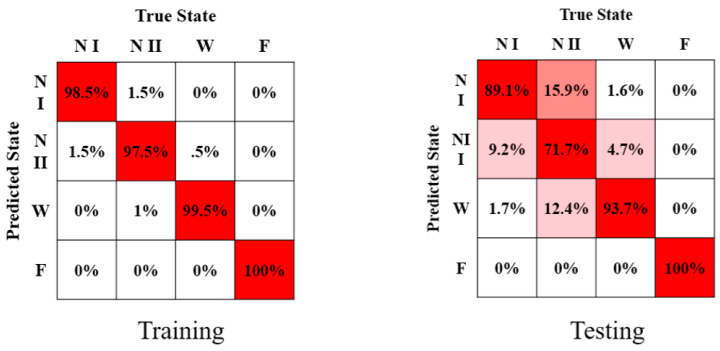


Figure 2.8: Confusion matrix from the RNN model for PdM (Wu et al., 2020)[30]

NI and NII states are the early states and normal working conditions of the machine, respectively. The matrix suggests that there are not much difference in those two conditions. Failure (F) and warning (W) states provide almost perfect accuracy (100% and 94%) for the algorithm. RNN provides higher accuracy than traditional learning algorithms.[30]

Another study by Killeen et al. demonstrated the usage of IoT for building a PdM algorithm.[12] Since the amount of data is enormous, the predictive model was divided into three sections referred to as an embedded system, the fog and the cloud. Embedded system contains perception systems including vehicle nodes to receive information. Server leader nodes are placed in the fog for local computations and the central management tool, which is referred to as the root node is in the cloud. The architectural representation of this IoT based system is given in Figure 2.9

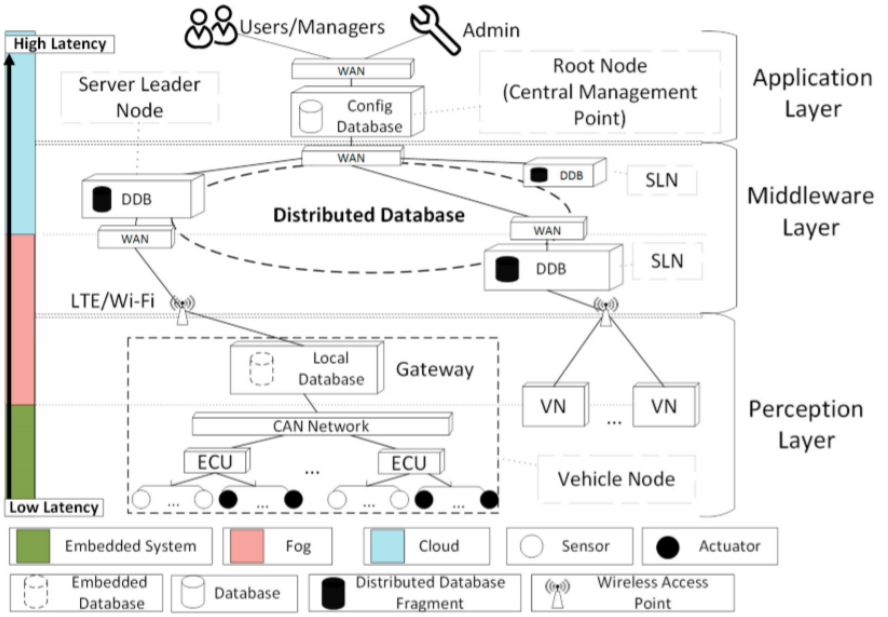


Figure 2.9: Architecture of IoT based PdM system. (Killeen et al., 2019)[12]

The perception layer sends information to the middleware layer for computation, which also exchanges data with the application layer for structural executions.

The successful application of neural networks is recognized in other studies as well. Namuduri et al. stated in a review paper for deep learning methods that long-short term memory-based neural network algorithms present superior prediction accuracy. [20] The algorithms that were tested in the review are logistic regression, fully connected neural network, support vector machine, ensemble and long-short term memory. [20] Those methods can be briefly explained as;

- **Logistic regression:** LR is a similar method to linear regression with a difference of binarising the response variable. It is basically estimating the parameters of a logistic function to model the response.[26]
- **ANN:** Artificial neural networks have a wide application in predictive maintenance. The structure is formed of layers of computational nodes,

which receives the output of the previous layer as the input. [30]

- **SVM:** The algorithm presented by Vapnik et al. proposes an efficient classification in high dimensional data by finding hyperplane(s) that has the largest margin from the nearest data point. [2]
- **Ensemble model:** Ensemble model is simply the usage of multiple algorithms rather than a single one to get better accuracy.
- **LSTM:** A type of recurrent neural networks, LSTM contains feedback loops in addition to its feedforward structure. It can be used to analyse an entire data set as well as analysing single data points. [30]

Comparing those algorithms proposed a clear result. LSTM demonstrated absolutely superior performance compared to other methods.[20] The results can be seen in Table 2.1. The area under the curve and precision are performance measures on the experiment.

Algorithm	Area under the curve	Precision
Logistic regression	0.990	0.596
Fully connected neural network	0.990	0.629
Support vector machine	0.990	0.593
Ensemble model	0.988	0.775
Long-short term memory	0.998	0.873

Table 2.1: Performance comparison of deep learning algorithms.[20]

2.5 Programming tools

Any kind of software that enables data analysis can be used for PdM. R, Python, Orange, MATLAB are few examples of such software. According to the webinar of pwc and mainnovation, the most used software tools for PdM are; MS Excel & MS Access(67%), CM software(40%), data software(33%), statistical software(18%), IoT(14%) and cloud(13%).[8]

Another important factor while implementing an algorithm if the system is identifying the type of data that will be used. It is essential to determine where to start collecting the data. Unnecessary data can slow down the process of missing out important data may result in an inaccurate algorithm. The types of data that are mostly used are; maintenance history(%73), condition(72%), usage(72%), data from other equipment(42%), environmental(29%), data from other companies(9%) and other types of data(7%).[8]

There are also several programmes and toolboxes that are designed specifically for RUL estimation and PdM planning. Microsoft's Azure also enables the design of a reliable algorithm and MATLAB has a PdM toolbox directly designed for applications. Similar products from various suppliers are available for industrial usage.

One example of such software is SensorCloud. It is a cloud-based data storage system which enables direct upload and storage of the data from the sensors to the cloud. It also provides a statistical analysis toolbox called MathEngine. MathEngine works with Python using Jupyter notebooks. The statistical analysis toolboxes of Python makes MathEngine a convenient tool for data analysis. The data will be sent to cloud instantaneously as the test is being done and can be used directly by the MathEngine.

SensorCloud provides documentation for CM and CBM using sensors and programming tools. Sample documentation for PdM based on machine performance in the oil industry and vibration analysis in factory equipment are provided within their guidelines. [25][24] Both documents indicate the possibility of predicting the downtime and finding the optimum performance by using CM equipment and simple programming tools.

Another example is MATLAB. MATLAB has a special toolbox for predictive maintenance applications. The predictive maintenance toolbox in MATLAB provides tools for simplifying commercial applications while keeping the reliability high.

3. Methodology

A well established PdM application requires large data sets from extensive periods. Additionally, data collection is a recursive action. The basic process for that is capturing data, analysing it, determining if new types of data are required, make new measurements if required and redoing it until obtaining adequate information. Since the duration of the project is limited, it is not possible to collect all the data within a short time and it is essential to prepare an effective plan of action. The project will have consisted of three main steps:

1. Creating a preliminary model based on the data available in the company in order to determine the most important parameters for the data collection part.

2. Building a test set by using the information obtained in the first step. This test will run for the entire project period and provide daily information about the status of the door.

3. Designing the final predictive model. By using the data from the tests, a fully operable algorithm can be generated. Optimization of the performance of this algorithm is the main goal of this step.

Several different design alternatives will be discussed in section 3.3 and the results of these will be compared in Section 4.2.

3.1 Preliminary model

The available data from the tests that were conducted in the second quarter of 2020 is very limited and does not provide enough information to design an accurate predictive model. Additional data is required for an improved design. The preliminary study is planned to have more insight into the failures of the doors and to determine the necessary parameters to be measured in the tests for improvements.

The available data is from a 4-day endurance test of a bus door. The test had to be stopped on the fourth day due to a mechanical failure in the door. Even though 4 day is a very short period to train a predictive model since the door broke down it provides considerable information on faulty working door systems. Signals acquired from the test are given in Figure 3.2. The data contains 12 channelled sensor information from the test. The sensors used in the test are an accelerometer in each door, measuring in x, y and z axes(3.2a), one strain gauge on each lower door arm (3.2c), 3 strain gauges on the door arm (-45^o, 0^o and $+45^o$ to the arm axis) and an amperemeter on the electric motor(3.2d). A placement scheme for the sensors is given in Figure 3.1. The exact places of the sensors are defined in Table 3.1.

Sensors	*Positions*						
	Lower door arm		Door frame		Electric motor	Upper door arm	
	Left	*Right*	*Left*	*Right*		*Left*	*Right*
Strain gauge*	0^o	0^o				$-45^o 0^o 45^o$	$-45^o 0^o 45^o$
Accelerometer**			XYZ	XYZ			
Amperemeter					+		

Table 3.1: Positions of the sensors for the preliminary endurance test.

**All the angles are relative to the movement axis of the sensor.*
***Orientations are relative to the axis of the accelerometer. Schematic representation can be seen in Figure 3.1.*

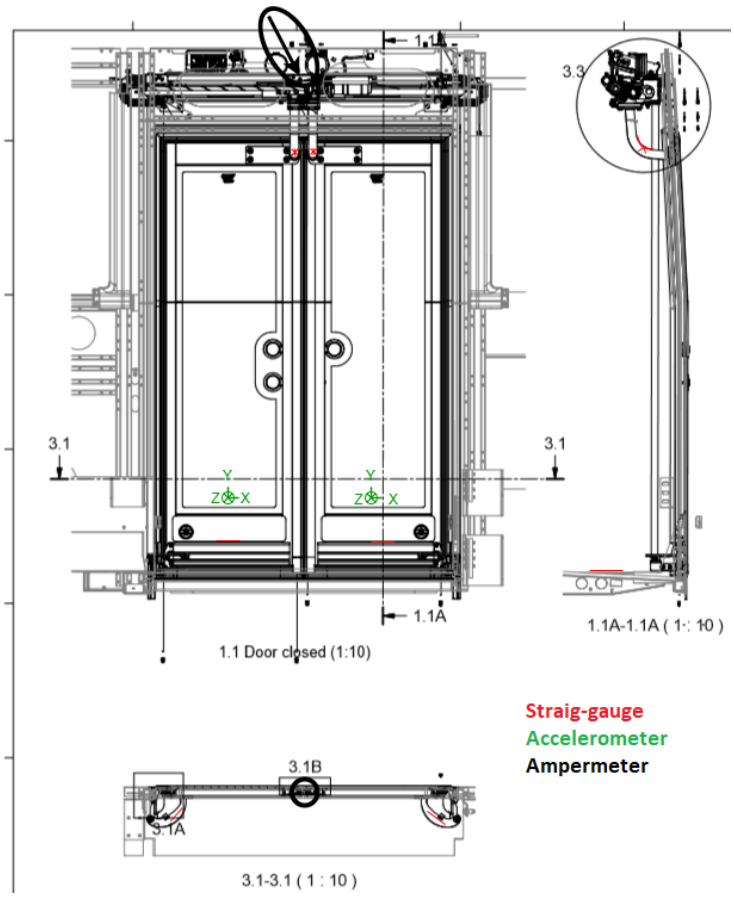
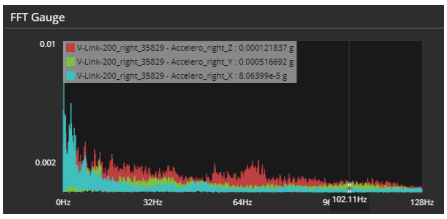


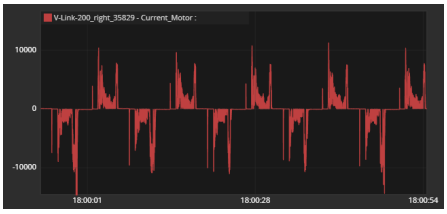
Figure 3.1: Sensor placement diagram of the first endurance test.

(a) Accelerometer signals.



(b) FFT of the accelerometer signals.

(c) Strain-gauge signals for lower door arms.



(d) Amperemeter signals for motor current.

Figure 3.2: Signals from the initial endurance test.

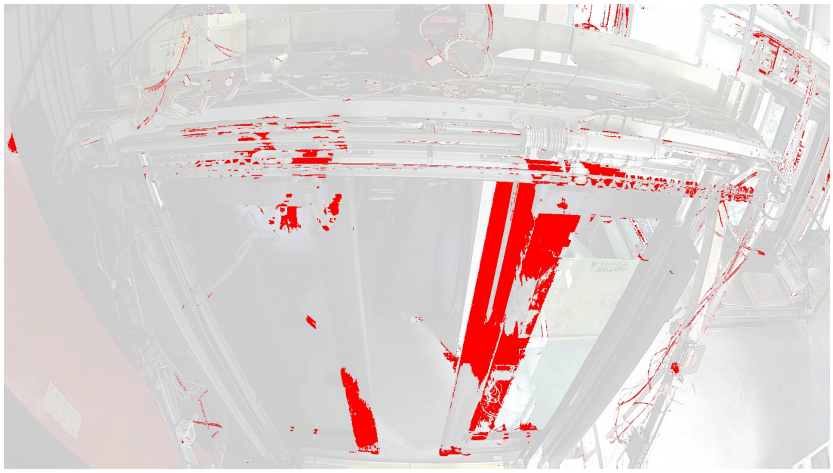
The sensors on the upper door arms had highly disturbed signal which was not providing useful information. The signal from Figure 3.3 was removed from the analysis and the connection with the sensor node was fixed for further tests. The data obtained from the sensors are stored in the cloud sensor data storage system, SensorCloud.

Figure 3.3: Noise dominated signal from the upper door arms strain gauge.

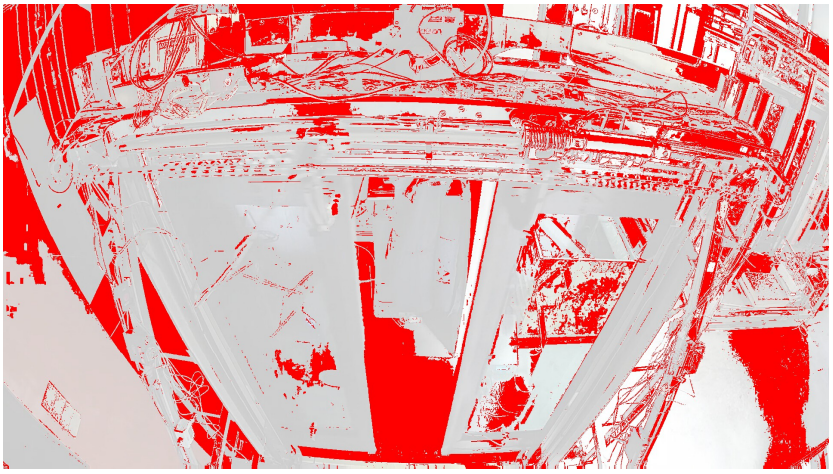
The change in the signals acquired from the sensors in time as the machine goes into the failure region is the feature of interest. The main goal is to find a pattern in the signals as the machine breaks down. For the accelerometer signal, it is also useful to measure spectral features by using FFT(Figure 3.2b).

The failure has occurred on the third night (between 12/07 and 13/07) of the test. However, it is not possible to detect the exact moment of the failure due to lack of data. The pictures of the test system, taken periodically are used to detect the estimated failure time of the system. Those pictures can be seen in Figure 3.4. It can be clearly seen from the pictures that the right arm

of the door mechanism was broken between 21.00 on 12/07/2020 and 08.00 on 13/07/2020.



(a) Picture from the test on 12/07/2020, 21:00.



(b) Picture from the test on 13/07/2020, 08:00.

Figure 3.4: Pictures of the door during endurance test.

Analysing the entire period in the data set is unnecessary and inefficient. It is also impossible to analyse the sensor data for longer periods. Therefore, analysing a signal for a length of 1 minute in each day was decided as adequate. The test started at 17.00 on 10/07/2020. The one-minute data samples are taken at 18.00 on 10/07 and 11/01, at 04.00 on 12/07 and at 12.00 on 13/07. The progression of the features, extracted from the data samples is an indicator of the failure. Even though this analysis can provide information solely on the failures of door mechanism arms, the algorithm can be extended for other types of failures once the data is acquired.

3.1.1 Learning algorithm

The available data is not enough to build a similarity or survival model for estimation. Similarity model requires a full life history of similar systems and survival model needs failure data from several machines. Since the only available data is from a 4-day test, degradation model will be used for RUL prediction. The flowchart for this algorithm that was generated with Python in MathEngine Notebooks of SensorCloud is given in Table 3.2.

Step	Notes
1. Uploading Data	Direct upload through SensorCloud
2. Get frequency-domain features	FFT's, Spectral Kurtosis and other spectral features.
3. Get time-domain features	Mean, Standard Deviation, Skewness and other types of time domain features of the signals.
4. Data smoothing	In order to eliminate the effect of noise in long term data, a moving average filter is applied to features.
5. Feature ranking	Monotonicity is the desired property for a health indicator. Features are ranked based on their monotonicity.
6. Feature selection	Features with higher monotonicity values are selected.
7. Normalization	Preprocessing for principal component analysis.
8. PCA	Dimension reduction for further analysis.
9. Generate health indicator and safety threshold	Monotonic PCA with a higher variance is selected to represent the health indicator. Threshold is predetermined based on company data.
10. Fitting algorithm	For the preliminary model, exponential curve fitting is used. Different methods will be investigated in posterior analyses.
11. RUL prediction	Based on the fitted model, RUL is predicted in every step by updating the parameters in the function.

Table 3.2: Flowchart of the preliminary prediction algorithm.

Accessing the data
MathEngine was selected due to its convenience in access to data. It is possible to directly upload the data from SensorCloud via an automatically generated code without downloading or transferring the data. This also enables continuous monitoring in a real door system.

Time-domain features
Mean, standard deviation, skewness, kurtosis, peak to peak amplitude, root mean square, crest factor, shape factor, impulse factor and margin factor are the time-domain features to be captured.

Those features can be expressed as;

Mean : The average of the signal.
Standard deviation : Amount of variation.
Skewness : Asymmetry in the distribution of the signal.
Kurtosis : The measure of "flatness" or "tailedness" of the distribution of the signal.
Peak to peak : The difference between the maximum and the minimum point

in the signal.
RMS : The square root of the average of the squared variables.
$RMS = \sqrt{\frac{avg(x^2)}{n}}$

Crest factor : The parameter that indicates the intensity of the peaks. Expressed as the rate of the maximum values to the RMS of the signal.
$CF = \frac{max(x)}{RMS}$

Shape factor : The ratio of the RMS to the mean of the signal.
Impulse factor : Ratio of the peak to the average of the signal.
$IF = \frac{max(x)}{avg(x)}$

Margin factor : Square of the impulse factor.

Frequency-domain features
Spectral features can give significant information about the system. FFT's generated from the data of accelerometers are used to investigate those features. Natural frequencies, spectral kurtosis, spectral mean, spectral standard deviation and spectral skewness are the features of interest.

Natural frequencies or eigenfrequencies of the signal are the vibration frequencies of the free oscillations. This feature is used only in vibration analysis. The rest of the spectral features are the same statistical properties described in time domain features, applied in the spectral distribution of the data.

Filtering and smoothing
A moving average filter with a window of 2 is applied to the features. Since the size of the data is low, the window is restricted to 2. A higher window would provide smoother data but in the cost of losing more points.
Smoothing the data is also required for the next step. Monotonicity will be used to rank and select the features of interest and noise in the data may disrupt the monotonicity calculations resulting in inaccurate RUL estimations.

Feature ranking and feature selection
Since it is expected that the machine goes into the failure region steadily by time, monotonic series from the features give further information about the status. Therefore, a monotonicity index was calculated to rank the features. According to that indexing, the features with highest monotonicity values are selected to be used in the prediction model.

Monotonicity implies the quantification of the monotonic trends in the serie. If a serie is monotonically increasing or decreasing, it will have a higher monotonicity index. The monotonicity index is calculated with equation 3.1.

$$monotonicity = \frac{1}{K}\sum_{a=1}^{K}\left|\sum_{b=1}^{L_a-1}\frac{sign(x_a(b+1)-x_a(b))}{L_a-1}\right| \tag{3.1}$$

[5]
The parameters and their meanings are given in Table 3.3

sign(x)	Signum function (returns 1 if x is positive and -1 if x is negative)
K	The number of systems (in this case, doors) monitored
$\mathbf{x}_i$	Vector of the feature measured on system j
$\mathbf{L}_i$	Number of data points on system j

Table 3.3: Parameters in Equation 3.1.

For the initial test analysis, K = 1 and L = 4(number of days). Features with a monotonicity value of 0.3 or higher will be included in the analyses.

Normalization
Since the features are not on the same scale, it is vital to normalize the features to avoid the dominance of features with a higher range. All the features are rescaled between -1 and 1.

Principal component analysis
It is possible to express the variance of the data with fewer components(sometimes even with 1 component). Thus, a PCA was done to find a principal axis that represents most of the variance for dimension reduction. For an accurate approximation, 95% confidence threshold is adequate. The principal components that provide more than 95% of the variance will be used for lifetime prediction. If only the first principal component explains more than 95% of the variance, then the prediction can be done with a single component.

Health indicator
The variable that represents the situation of the machine is called "health indicator". It is used to fit a prediction function for RUL estimation. A threshold is determined by the company, based on their knowledge and as the time between the day of measurement and the predicted day of overshooting of the threshold is the remaining useful life of the machine.
The health indicator is the combination of principal components (or only the first component, if it has a variance higher than 95%) that have a monotonic trend. A monotonic set is required for health indication since the health of a machine can not incline to opposite directions in time.

Algorithm
For the preliminary model, a basic exponential function,

$$y(t) = a * e^{b*t} + c \tag{3.2}$$

will be fitted to the data. The parameters a, b and c are updated in every iteration. With every newly acquired data, the coefficients are recalculated to make a better prediction. The accuracy of the prediction increases as the machine moves towards the failure point.

Estimating the remaining useful life
The safety threshold is the point where the machine is not operable anymore. It can be determined based on the prior studies on the doors or a rough estimation based on the parameters and the know-how of the company. The time required for the machine to surpass the threshold is the remaining useful life of

the machine. The basic approach to that is;

$$RUL = \frac{\ln(\frac{y_t - c_{opt}}{a_{opt}})}{b_{opt}} - t_c \tag{3.3}$$

Where y_t is the threshold value for the health indicator, *opt* subscript implies the optimum parameters of Equation 3.2 and t_c is the current day.

3.1.2 Outcomes

The results of the initial analysis can be interpreted in several ways. First of all, the correlation matrix explains the dependencies of the sensors. This can be used to eliminate the unnecessary equipment in the upcoming tests and determine additional measurements. The correlation matrix can be seen in Figure 3.5.

Figure 3.5: Correlation matrix of the parameters for the initial endurance test.

It was expected that the right lower strain-gauge would exhibit a different value set than the one on the left lower door arm since the failure occurred on the right-upper door mechanism arm. However, according to the correlation matrix, the left and right tension values are 100% correlated. This indicates that either both doors are affected by the fracture on the upper right door mechanism arm or more likely, both of the doors were not affected by the fracture. In both cases, it can be deduced that it is essential to determine a new way to monitor the situation of the upper door mechanism arm. This problem can be solved by placing strain gauges on the upper door mechanism arms on the new test set.

Motor current and vibration on the X direction (the direction of the door movement) are 80% correlated. Since the opening force is directly related to the motor current and the vibration is related to the force on the movement axis, it is an expected result. However, this does not imply that one of the parameters is redundant. Motor current can also be used to identify electrical failures, in addition to mechanical failures thus it is necessary to measure. Vibration on X direction can also be used to determine the robustness of the bolts and other connection elements. Both parameters can not be removed due to their ability to identify different types of failures but for mechanical failure on the door arm, it is possible to use information from only one of them, to reduce the computational load.

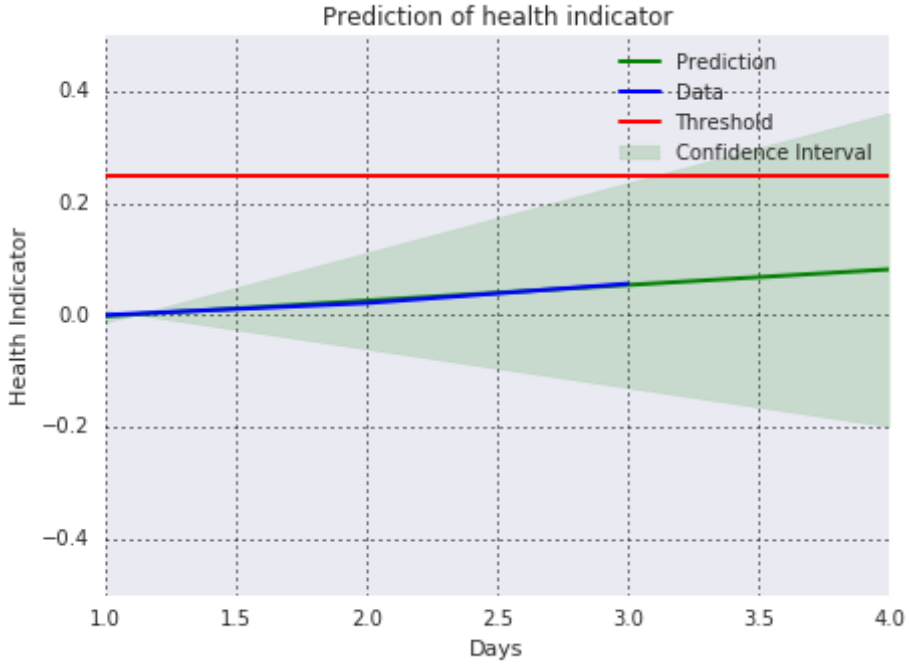
Vibration on Z and X directions are also highly correlated, 90%. One of them can be eliminated. However, their variance in different types of failures must be evaluated before deciding which one to eliminate in order to ensure the accuracy of the model.

For this study, since only a single type of failure is the response of interest, for computational convenience vibration X will not be used in the further analysis after validating its redundancy. However, the measurements will be taken for further studies.

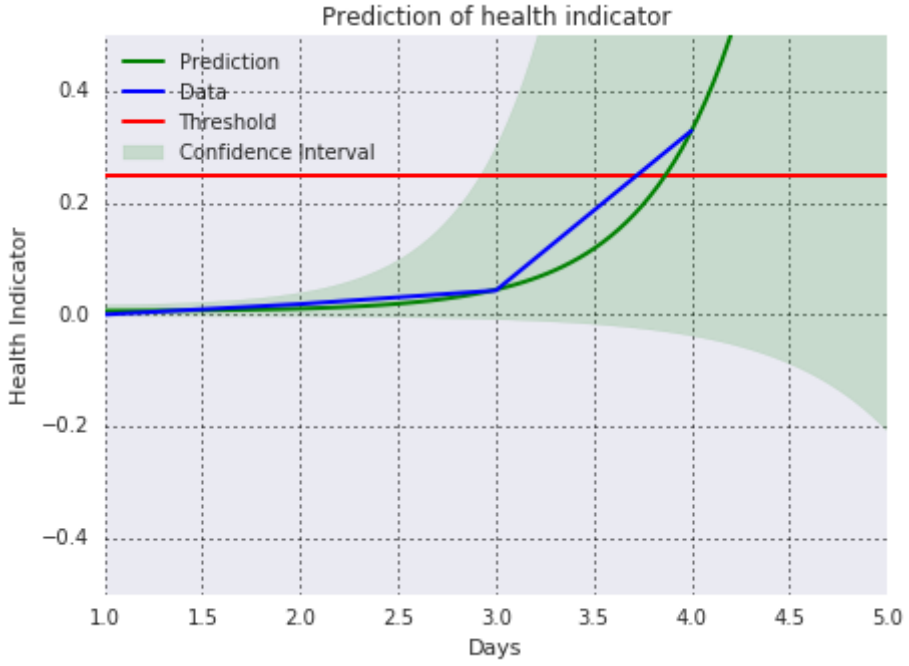
After a PCA, it was observed that the first principal component explains 80% of the variance. This component can be used for a lifetime prediction. It is known that the failure occurred between the 3rd and the 4th day so a threshold between those days can be selected. With the extracted information, a basic prediction can be made to see how the program performs. The predictions on the 3rd and 4th date can be seen in Figure 3.6. The estimated RUL's for there predictions are 2 days (3.6a) and -0.2 days (3.6b). Negative remaining life is an accurate prediction for the 4th day since the door had broken down after the 3rd day. Considering the amount of data used to train, the basic model provided a satisfying result.

For this prediction, pythons curve_fit function was used. The parameters of equation 3.2 were updated for each day and the prediction is repeated. This is a simple algorithm to estimate the RUL and results of the complex algorithms will be discussed in the following sections. Nevertheless, this basic algorithm provided partially adequate results for very limited data.

It is not expected to predict with 100% accuracy. A confidence interval of the estimation can be calculated with the results of curve_fit() function to be demonstrated on the plot. Even though the confidence interval relatively broad for a prediction model, since this is only the test algorithm it is considered acceptable.



(a) Health indicator prediction based on 3-day-data.



(b) Health indicator prediction based on 4-day-data.

Figure 3.6: Machine health predictions of the preliminary model.

Since the behaviour of the door did not change significantly for the first 3 data points, the prediction is a straight line. Therefore its confidence interval is narrower than the prediction made in the 4th day. The exponential function is expected to fit the data better as the number of sample points increase. For only 3 samples, a straight line is an expected outcome. After the condition changes seriously on the 4th day, the fit is more descriptive. However, the steep ascent in the health indicator caused uncertainty in the prediction, extending the confidence interval and decreasing the prediction accuracy. The accuracy is expected to increase with more samples therefore in the second test run, after a certain time the model will have a narrower confidence interval for failure prediction.

3.2 Data collection

The initial endurance test provided valuable results to build a new test set with the purpose of acquiring more informative data. This is the recursive step indicated in Figure 2.2 of Section 2.1.
A moving average filter was planned to be used to reduce the effect of the noise in the signal. However, since the number of data points is already low, a window of only 2 could be used in this step. This resulted in a data set which is influenced significantly by the noise. In addition to that, the data could not be very informative due to the lack of measurements. The features were ranked based on their monotonicity as explained and analysed after normalization. Similarly, the monotonic behaviour of 4 point data can not be reliable. All these problems indicate that this model can only be used as a template to the real predictive model. The most important outcome of the initial test is that it indicated the improvement points so that in the next step, a better test set can be built and more reliable results can be obtained.

In order to obtain more informative data, a test system for another PSD was constructed. Placement and types of some sensors are changed based on the results from the initial test. The number of strain gauges on the upper door arms is reduced to one. The stress on the perpendicular axis was relatively minor, compared to the axial direction, thus the strain-gauges that were placed $+45^o$ and -45^o were removed. The amperemeter was changed with a shunt resistor due to its lack of accuracy on current measurements. Shunt resistor provided considerably more reliable results in trial measurements. The temperature sensor was installed in the bearing and the electric motor. Temperature measurements can give important information about different types of failures. Finally, a strain-gauge was placed on the vertical axis of the upper door mechanism arm, which fractured on the first test, to obtain more information about the part. The accelerometers on both doors are not altered.

The test system contained two accelerometers in both doors, one strain-gauge in both lower door arms, one strain-gauge on both upper door arms, one strain-gauge on both upper door mechanism arms, a temperature sensor in left bearing, a temperature sensor in the electric motor and a shunt resistor to measure the motor current. The positions of the sensors for the main endurance test are defined in Table 3.4 and are shown in Figure 3.7.

Sensors	*Positions*									
	Lower door arm		Door frame		Motor	Top door arm		Bearing	Mechanism arm	
	Left	*Right*	*Left*	*Right*		*Left*	*Right*		*Left*	*Right*
Strain gauge	0^o	0^o				0^o	0^o		90^o	90^o
Accelerometer			XYZ	XYZ						
Shunt resistor					+					
Temperature					+			+		

Table 3.4: Positions of the sensors for the main endurance test.

**All the angles are relative to the movement axis of the sensor.*
***Orientations are relative to the axis of the accelerometer. Schematic representation can be seen in Figure 3.7*

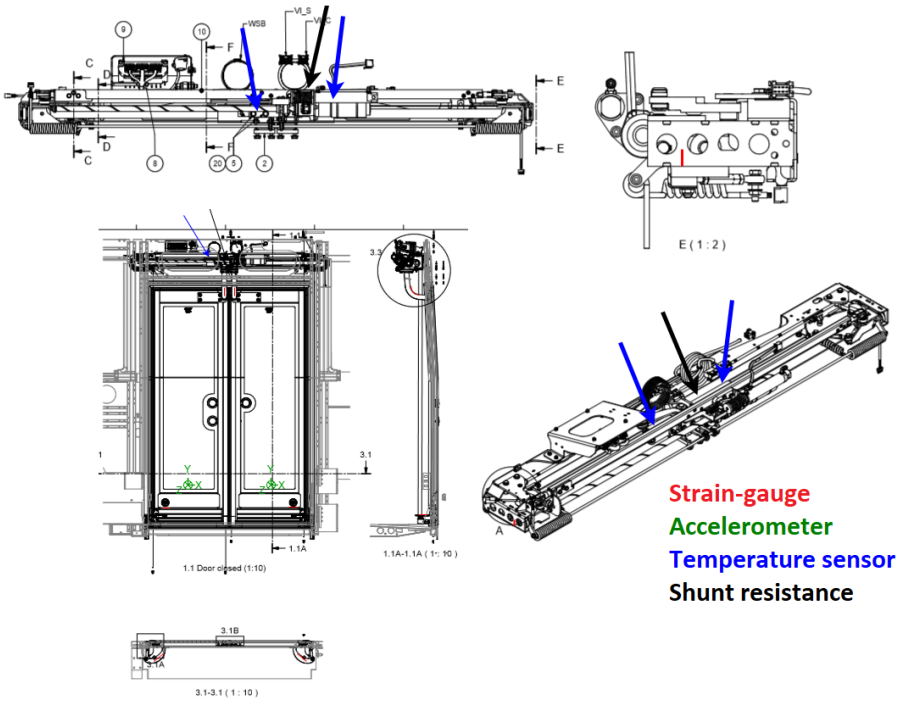


Figure 3.7: Sensor placement diagram for the main endurance test.

The optimum number of sensors that are necessary for a complete model is unknown. Before deciding which sensors to eliminate and which data to collect, it is essential to understand the failures that can occur in a bus door. Current measurements can be irrelevant to connection element failures but it can give information about the electrical failures. Moreover, the types of sensors that can be used in a real bus are limited. It is not possible to implement strain gauges on every risk spot on a real bus so a correlation between variables must be observed to connect the relations between sensors. Therefore a broader test set is designed to be examined.

Similar to the initial test, the sensors are connected to the cloud database to send instantaneous signals for analysing. The test is planned to run continuously.

In general, the tools that are used in the project can be summarized as;

Measurement tools
-Strain gauges
-Accelerometers
-Temperature sensors
-Current sensors

Analysis tools
-SensorCloud
-MathEngine
-MATLAB

The final test has started at 14:12, 6th of October, 2020.

The temperature measurements have a transient period and a stationary period. Both motor and bearing temperatures increase after the test starts. In order to ensure that the analysis is unbiased, the samples must be taken after this transient period. The convergence of the signals for temperature sensors can be seen in Figure 3.8

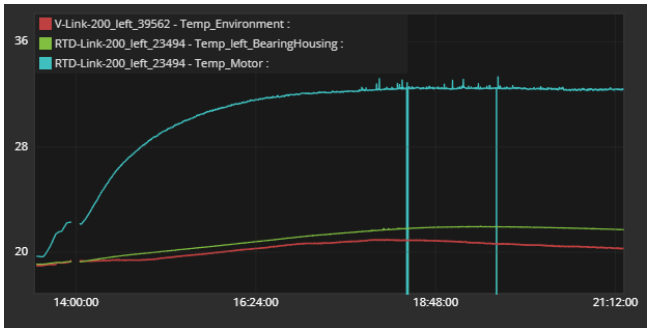
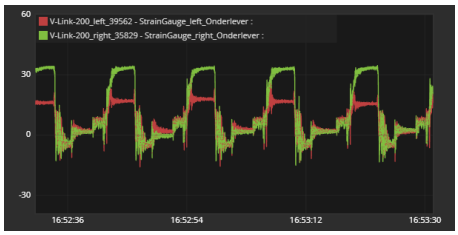


Figure 3.8: Temperature readings for the transient zone.

The environment temperature is approximately 19.5^o C and assumed to be constant through the test. The motor temperature (blue line) starts from 19.5^o C and increases up to 33^{o}C over time. After that, it oscillates between 34 and 32, correlated to the hour of the day. The bearing temperatures increase with time similarly to the motor temperature, but with a smaller difference. Similarly, the bearing temperature reaches a stationary point with an offset of approximately 1.5^{o}C to the environment temperature. The samples to analyse can be taken when that point is reached.

The other signals from the sensors are given in Figure 3.9 where figs. 3.9a–3.9i represent the signals from strain gauges in lower door arms, upper door arms, door mechanism arms, roller mechanism, accelerometers on x, y and z axes, motor current measurements and voltage measurements, respectively.
On the plots with two lines, the green lines represent the signals from the right node (for right door) and the red lines represent the signals from the left node

(for left door).



(a) Strain gauge signals from lower door arms.

(b) Strain gauge signals from upper door arms.

(c) Strain gauge signals from door mechanism arms.

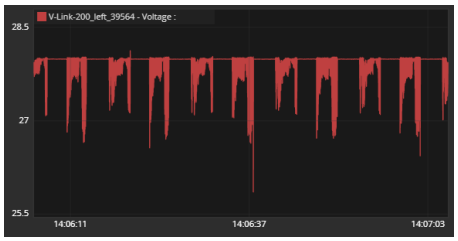
(d) Strain gauge signals from roller mechanism.

(e) Accelerometer signals on X axis.

(f) Accelerometer signals on Y axis.

(g) Accelerometer signals on Z axis.

(h) Motor current signal.



(i) Voltage signal.

Figure 3.9: Sensor signals from the final endurance test.

Similar to the primary test, the data samples are taken once in every 24 hours to reduce the effect of the temperature change since strain gauges are extremely sensitive to temperature differences. Due to the amount of data, the interval for sampling is 36 seconds, slightly higher than half of the samples in the primary test. A full cycle takes approximately 12 seconds, meaning that in each sample there are exactly 3 opening and closing actions. In addition to the time adjustment, a synchronization operation is introduced to ensure that the collected data are from the exact time stamps of the door cycles. All the data for the 60 day period start from the beginning of an opening cycle and end in the 3rd closing cycle after the sampling starts. The data set contains information from the test for 60 days, 36 seconds each. The total time of the data is 2160 seconds. The sampling rate is 256 Hz meaning that 7776 data points are collected per day and 466.560 data points in total.

3.3 Improved model

With the larger data set obtained from the second endurance test. A final algorithm is designed for advanced prediction. The initial approach uses the preliminary model as a base and builds the improvements on it. Similarly, the model is updated with every data acquired and RUL is predicted according to that.

The exponential regression algorithm is almost the same as the one used in the preliminary analysis. MathEngine notebooks of SensorCloud are used to generate an exponential regression model based on the simple least-squares method. The script of the initial algorithm is used with minor modifications. The steps of the analysis are exactly the same as in Table 3.2. However, in this run, feature ranking and selection is more reliable since most of the data is still in the model even after the moving average filter and monotonicity index represents more sensible results. With the larger data, it is possible to observe the effects of the progression of features on the situation of the door. However, for performance comparison, two other algorithms are introduced at this step: exponential degradation and artificial neural network.

3.3.1 Exponential degradation

Gathering larger and more informative data enables a new opportunity. In addition to the Python script, MATLAB's predictive maintenance toolbox is used for the validation of the results. The toolbox uses a degradation function that updates the parameters based on variables extracted from probability distributions of the parameters. Thus it is expected to obtain different results to a certain extent but converge to the same point as the time increases.

The steps in this approach are similar to the example study, *Wind Turbine High-Speed Bearing Prognosis* of MATLAB's documentation on Predictive Maintenance Toolbox.[17]

Algorithm

The degradation function that MATLAB uses is based on the paper of Gebraeel(2006) and his algorithm to define distributions with degradation patterns. [7] The function is expressed as in Equation 3.4.

$$S(t) = \phi + \theta(t) * e^{\left(\beta(t)*t+\epsilon(t)-\frac{\sigma^2}{2}\right)} \tag{3.4}$$

The parameters used in this equation are defined in Table 3.5

$\boldsymbol{\phi}$	Constant in the system.
$\boldsymbol{\theta(t)}$	Random parameter which is drawn from a *longnormal* distribution with a mean and standard deviation that are updated over time.
$\boldsymbol{\beta(t)}$	Random parameter which is drawn from a *Gaussian* distribution with a mean and standard deviation that are updated over time.
$\boldsymbol{\epsilon(t)}$	Noise factor used in the equation for increased accuracy. Drawn from a normal distribution with zero mean and time dependent variance.
$\boldsymbol{\sigma(t)^2}$	Variance of ϵ.

Table 3.5: Parameters for Equation 3.4.

3.3.2 Artificial neural network

An artificial neural network or shortly, ANN is one of the most reliable methods in time series analysis. [30][20] A simple neural network is used to capture the time-dependent pattern in the experiment data and to predict the progression of the health indicator of the door. MATLAB's neural network toolbox is used to build the network. The tool that will be used is a Nonlinear Autoregressive Neural Network (NAR). This is a type of recurrent neural network, meaning that the algorithm uses 'd' numbers of past values of a serie 'y' to determine the future values of the serie. The algorith can be mathmematically expressed as;

$$y(t) = f(y(t-1), y(t-2), ..., y(t-(d-1)), y(t-d)) \tag{3.5}$$

The parameters in Equation 3.5 are $\boldsymbol{d}$, time delay in the network; $\boldsymbol{t}$, discrete time steps and $\boldsymbol{y(t)}$, value to be predicted.

The number of hidden layers and the delay are updated for each sensor for optimizing the prediction. There is a single output layer that gives the prediction of the series y which was mentioned as the health indicator before. A schematic representation of the network can be seen in Figure 3.10.

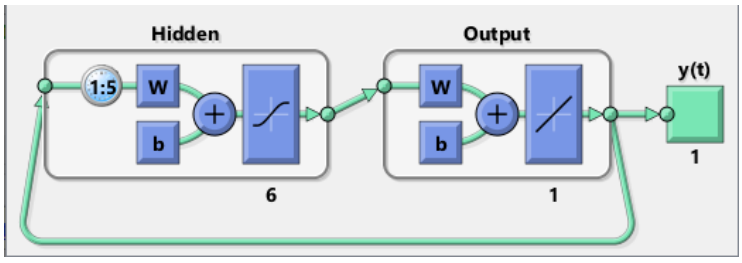


Figure 3.10: NAR Neural Network

3.3.3 Preprocessing

The preprocessing part is the same in all three algorithms. The extracted features does not differ since the preprocessor is unchanged, meaning that it uses the same principal components, or consequently the same health indicator data.

The initial step is to visualise the data. Figure 3.11 represents the data from the accelerometer in time-domain. For the vibration data, frequency-domain features can be even more useful. To express the data in frequency-domain, the FFT generated from the left door accelerometer on x-axis data can be seen in Figure 3.12.

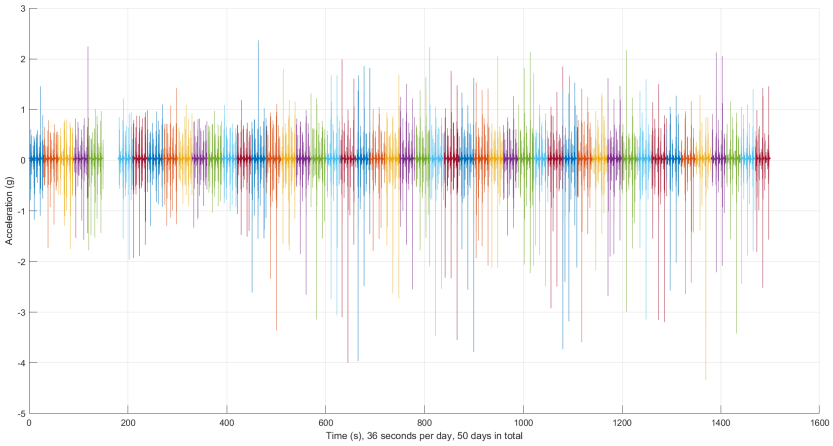


Figure 3.11: Left door accelerometer (X axis) data.

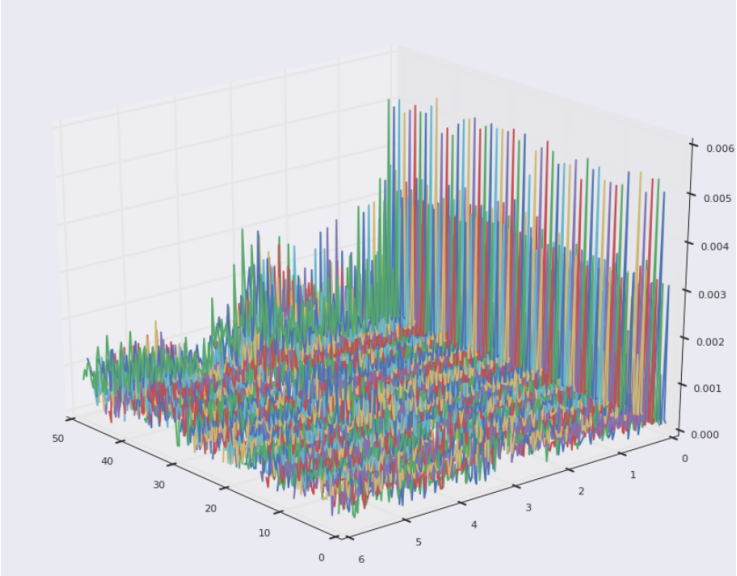


Figure 3.12: FFT plot of the accelerometer (X axis).

The first feature to be extracted is the spectral kurtosis. Spectral kurtosis indicates non-Gaussian behaviour of the data in the frequency domain. The changes in the spectral kurtosis of the data over time can indicate a degradation in the signal, which will provide useful information for the condition prediction. The Spectral Kurtosis - Frequency - Time plot can be seen in Figure 3.13.

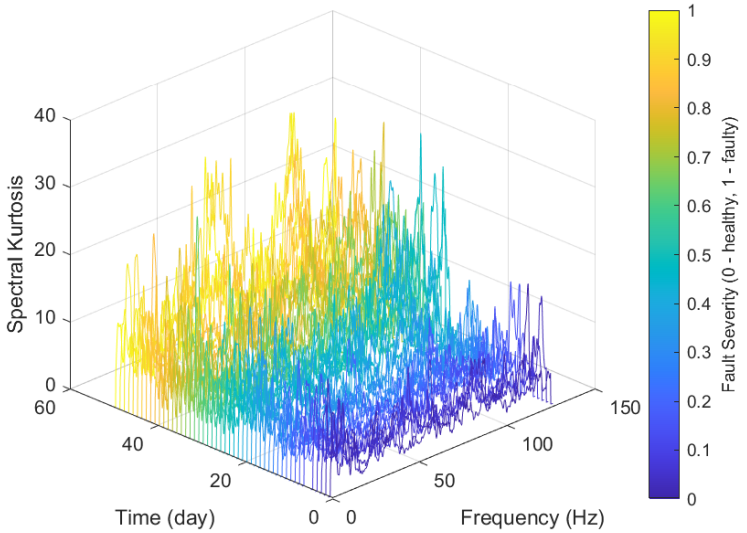


Figure 3.13: Spectral kurtosis plot of the accelerometer data. (X-axis)

In addition to spectral features, time-domain features are extracted for further processing. Since the data from the 5th day is missing, the value for the features of the 5th day is filled with the mean of the values from 4th and 6th day. A moving average filter with a window size of 5 is applied to the features to reduce the random noise in the data. The feature table for the first 30 days after smoothing operation can be seen in Table 3.6.

	Date	1 Mean	2 Std	3 Skewness	4 Kurtosis	5 Peak2Peak	6 RMS	7 CrestFactor	8 ShapeFactor	9 ImpulseFactor	10 MarginFactor	11 Energy	12 SKMean	13 SKStd	14 SKSkewness	15 SKKurtosis
1	06-Oct-2020 18:00:00	-54.4395	76.8425	-0.5698	1.3658	194.6494	94.1682	0.0975	1.5568	0.1518	0.0025	6.8112e+07	3.1356	3.5360	1.1153	3.3431
2	07-Oct-2020 18:00:00	-48.8384	74.2121	-0.7674	1.6757	193.3623	88.8954	0.0928	1.6575	0.1534	0.0029	6.0912e+07	3.3074	3.4437	0.8975	2.7969
3	08-Oct-2020 18:00:00	-51.4827	75.1209	-0.6754	1.5425	193.1423	91.1344	0.0963	1.6150	0.1549	0.0028	6.4014e+07	3.3960	3.3852	0.7520	2.4739
4	09-Oct-2020 18:00:00	-51.4011	74.8842	-0.7083	1.5825	193.7100	90.8759	0.0875	1.6435	0.1425	0.0026	6.3599e+07	3.2417	3.2743	0.8221	2.6836
5	10-Oct-2020 18:00:00	-55.0658	75.0188	-0.6372	1.5031	196.2064	93.2625	0.0734	1.6097	0.1190	0.0021	6.7116e+07	3.2012	3.2326	0.8538	2.8044
6	11-Oct-2020 18:00:00	-57.5501	76.0351	-0.6134	1.4721	199.1921	95.5816	0.0623	1.5967	0.1009	0.0018	7.0660e+07	3.1692	3.1729	0.8622	2.8525
7	12-Oct-2020 18:00:00	-60.1801	77.6739	-0.6246	1.4858	203.7587	98.4778	0.0457	1.6020	0.0750	0.0014	7.5236e+07	3.1432	3.0306	0.8144	2.7785
8	13-Oct-2020 18:00:00	-66.1538	80.6386	-0.5346	1.3593	208.2442	104.4290	0.0312	1.5604	0.0494	8.3756e-04	8.4513e+07	3.1761	3.0338	0.8861	3.1500
9	14-Oct-2020 18:00:00	-66.4472	81.3956	-0.6248	1.4982	212.6135	105.2136	0.0078	1.5911	0.0126	2.3678e-04	8.5694e+07	3.3148	3.3339	0.9585	3.2951
10	15-Oct-2020 18:00:00	-69.7546	82.9955	-0.6238	1.4934	216.9252	108.4986	-0.0140	1.5607	-0.0231	-3.5960e-04	9.0738e+07	3.4513	3.4555	0.9372	3.2508
11	16-Oct-2020 18:00:00	-69.6825	83.4857	-0.6598	1.5211	215.6039	108.8140	-0.0236	1.5668	-0.0374	-5.6245e-04	9.1241e+07	3.5730	3.7337	1.0467	3.7675
12	17-Oct-2020 18:00:00	-69.9179	83.8092	-0.6662	1.5201	215.3026	109.2036	-0.0276	1.5672	-0.0434	-6.4669e-04	9.1865e+07	3.7699	3.9868	1.0416	3.7215
13	18-Oct-2020 18:00:00	-70.2011	83.8950	-0.6638	1.5146	215.4183	109.4492	-0.0298	1.5644	-0.0469	-6.9556e-04	9.2289e+07	3.8167	4.1304	1.0300	3.6318
14	19-Oct-2020 18:00:00	-68.7861	84.0674	-0.7420	1.6157	218.3392	108.6768	-0.0382	1.5837	-0.0605	-8.8766e-04	9.0900e+07	3.7062	4.0951	1.0547	3.6314
15	20-Oct-2020 18:00:00	-70.1783	85.5952	-0.7160	1.5629	221.2948	110.7313	-0.0346	1.5787	-0.0544	-7.7311e-04	9.4261e+07	3.5281	3.7787	1.0288	3.6090
16	21-Oct-2020 18:00:00	-71.3534	87.0759	-0.6508	1.4850	222.5581	112.6206	-0.0159	1.5778	-0.0253	-3.7911e-04	9.7613e+07	3.6817	3.7902	0.9933	3.5116
17	22-Oct-2020 18:00:00	-69.7347	88.3996	-0.7067	1.5702	228.0868	112.6668	-0.0057	1.6194	-0.0091	-1.3084e-04	9.7687e+07	3.7655	3.5571	0.8315	2.8944
18	23-Oct-2020 18:00:00	-69.0186	89.1767	-0.7288	1.6000	230.0802	112.8193	9.5727e-04	1.6376	0.0015	2.3284e-05	9.7945e+07	3.7494	3.4139	0.8398	2.9535
19	24-Oct-2020 18:00:00	-67.0498	88.4231	-0.7802	1.6811	229.3385	111.0170	9.0280e-04	1.6591	0.0011	4.5676e-06	9.4965e+07	3.9692	3.4723	0.8608	3.0784
20	25-Oct-2020 18:00:00	-69.4198	89.1931	-0.7066	1.5818	229.2225	113.1283	0.0059	1.6386	0.0096	1.3656e-04	9.8891e+07	4.2320	3.6812	0.7927	2.8415
21	26-Oct-2020 18:00:00	-70.4839	86.8242	-0.6519	1.5054	222.0713	111.9997	-0.0049	1.5987	-0.0054	-6.2051e-05	9.6996e+07	4.4528	3.8252	0.7657	2.7849
22	27-Oct-2020 18:00:00	-69.5604	84.3215	-0.6831	1.5366	215.8473	109.5140	-0.0259	1.5860	-0.0368	-4.8300e-04	9.2655e+07	4.3998	3.8298	0.8409	3.0384
23	28-Oct-2020 18:00:00	-68.7669	81.8667	-0.7012	1.5727	209.7972	107.0609	-0.0382	1.5671	-0.0574	-8.5017e-04	8.8976e+07	4.3878	3.9842	0.9433	3.2882
24	29-Oct-2020 18:00:00	-66.9244	78.5213	-0.7633	1.6791	203.2949	103.2955	-0.0586	1.5544	-0.0896	-0.0014	8.3242e+07	4.5960	4.1352	0.8807	3.1197
25	30-Oct-2020 18:00:00	-65.1854	75.7143	-0.7324	1.6311	196.1901	100.0048	-0.0497	1.5473	-0.0747	-0.0011	7.8574e+07	4.5082	4.0543	0.8567	3.0586
26	31-Oct-2020 18:00:00	-62.2550	71.6407	-0.7273	1.6292	186.9526	95.0231	-0.0515	1.5386	-0.0772	-0.0012	6.9971e+07	4.5267	3.9232	0.7783	2.8769
27	01-Nov-2020 18:00:00	-60.3815	70.1670	-0.7272	1.6298	183.6841	92.6665	-0.0455	1.5443	-0.0687	-0.0011	6.6394e+07	4.4615	3.9069	0.8102	2.9365
28	02-Nov-2020 18:00:00	-58.2587	68.7228	-0.6970	1.6017	180.9836	90.1727	-0.0254	1.5537	-0.0390	-6.4937e-04	6.2611e+07	4.2733	3.7651	0.7609	2.7319
29	03-Nov-2020 18:00:00	-58.8882	67.3408	-0.6761	1.5647	177.0775	89.5196	-0.0383	1.5249	-0.0565	-9.3014e-04	6.1725e+07	4.2325	3.6605	0.6560	2.4057
30	04-Nov-2020 18:00:00	-58.1195	66.8284	-0.6744	1.5625	175.7215	88.6372	-0.0343	1.5312	-0.0507	-8.5881e-04	6.0570e+07	4.1516	3.6585	0.6658	2.3850

Table 3.6: Feature table for the left door accelerometer, x axis data.

The monotonicity indices for all features are calculated to select the most

useful data. A threshold of 0.3 was selected to delete the features with lower monotonicity index. Figure 3.14 show the sorted monotonicity indices of the features, sorted from highest to lowest. According to this plot, *Crest Factor, Kurtosis, Impulse Factor* and *Margin Factor* will be used to build the predictive model.

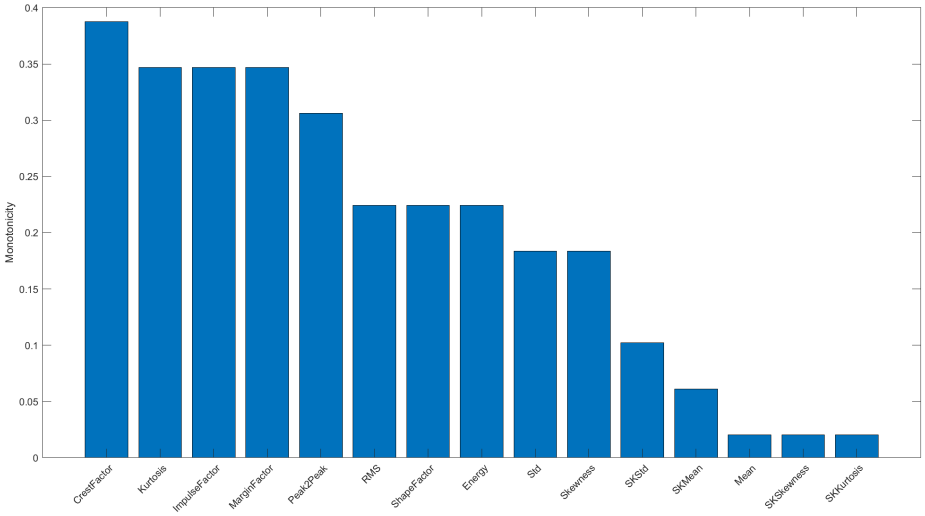
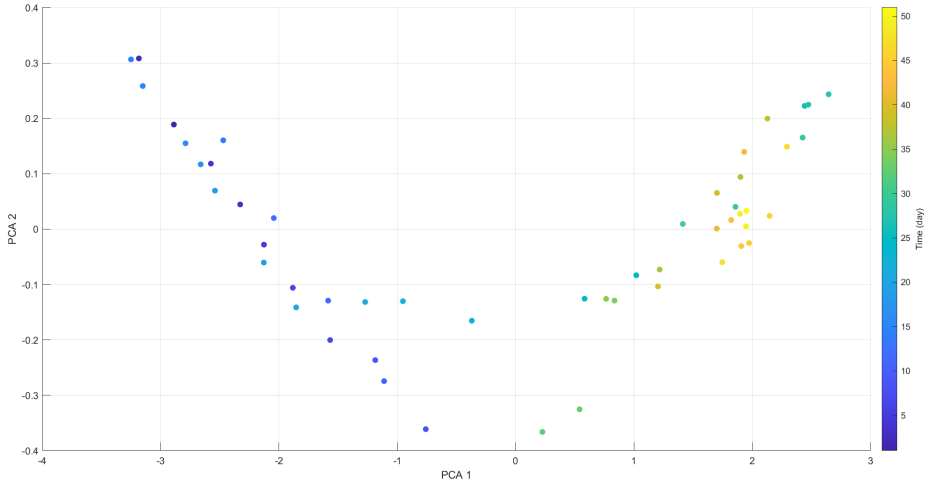
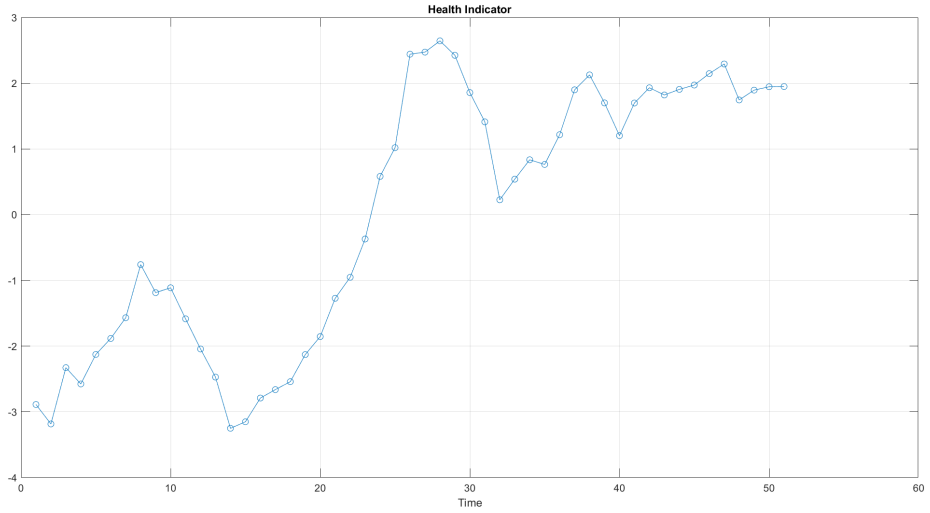


Figure 3.14: Sorted bar plot of monotonicity indices of the extracted features for the left door accelerometer, x-axis data.

The last step before the prediction is dimension reduction. Principal component analysis on the selected features gives the principal components which express most of the variance, thus enables the reduction of the number of variables used in the predictive model. The first principal component represents more than 90% of the variance, meaning that it can be used as the health indicator of the door. The results are given in Figure 3.15. Subfigures 3.15a and 3.15b represent the plot if the first two principal components and the health indicator.



(a) First two principal components.



(b) Health indicator of the door for left door accelerometer, x axis.

Figure 3.15: Results of the PCA.

After determining the health indicator, the three aforementoned algorithms are used to generate a fit and to predict the RUL of the machine.

4. Results and Discussions

The main interest was on the effect of the algorithm on accuracy and the correlations between the parameters. The study provided satisfactory results for both of the questions and the results are interpreted below.

The test was done with 10 different sensors, meaning that a separate prediction can be made for each component, resulting in 10 different predictions. To explain the interpretation of the results, the prediction based on the left door accelerometer data on the X-axis will be investigated. The rest of the predictions and results are given in Appendix B The comparison, on the other hand, will be done with the information gathered from the entire data collection setup.

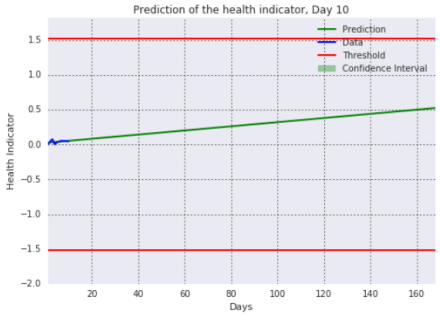
4.1 Prediction algorithms

Although it is possible to use any type of learning algorithm, experimenting with every algorithm is unnecessary and inefficient at this stage of the project. Three algorithms are proposed for this study and the most feasible design will be selected among the alternatives. The first algorithm is exponential regression. Since it is one of the most simple methods that can be used in machine learning, it is possible to observe the prediction quality even in the most basic solution. The second alternative is a recurrent neural network, which was mentioned as one of the most efficient methods in several studies inspected in Section 2.4[30][20]. The third algorithm is an exponential degradation based on the PDF of the data.

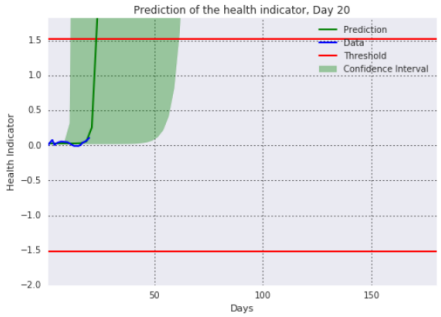
The doors are expected to work for approximately 1.500.000 cycles. The average number of cycles performed during the test in one day is 8500, meaning that the test should run for approximately 180 days before the door fails to operate. Since the exact value of the safety threshold is not yet known for the company, the values are adjusted to fit this criterion. A performance evaluation can be done between the algorithms afterwards.

4.1.1 Exponential regression

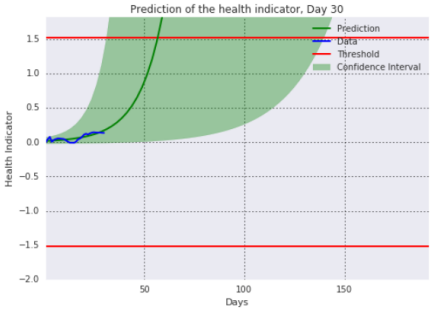
The threshold is determined based on the 50 day data of the test. The 50 day period is used as the training set, to train the predictive model and the remaining 10 days are used to test the prediction accuracy. The prediction plots taken from 10th, 20th, 30th, 40th and 50th days are given in figs. 4.1a–4.1e.



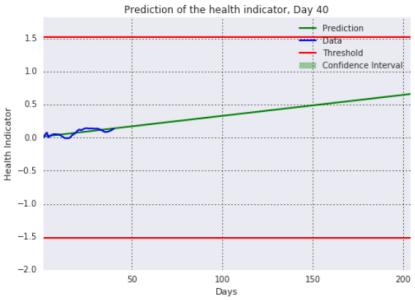
(a) RUL prediction on 10th day.



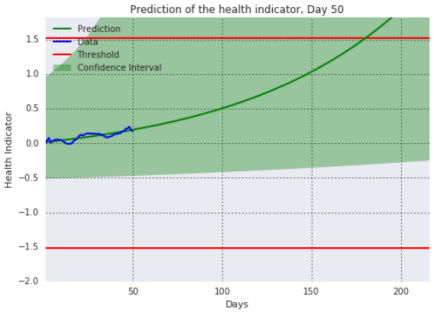
(b) RUL prediction on 20th day.



(c) RUL prediction on 30th day.



(d) RUL prediction on 40th day.

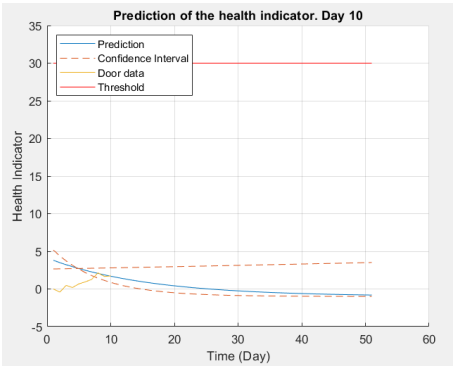


(e) RUL prediction on 50th day.

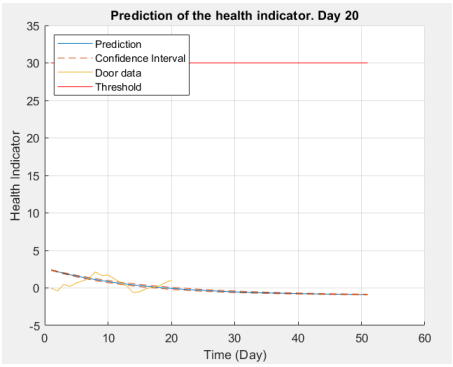
Figure 4.1: Predictions of the regression algorithm.

4.1.2 Exponential degradation

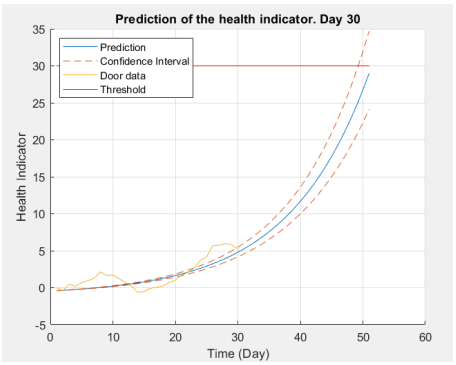
Similarly to the regression model, degradation model predicts the RUL for days 50 to 60 based on the 50-day data. The predictions for the same time periods with the degradation model is represented in figs. 4.2a–4.2e.



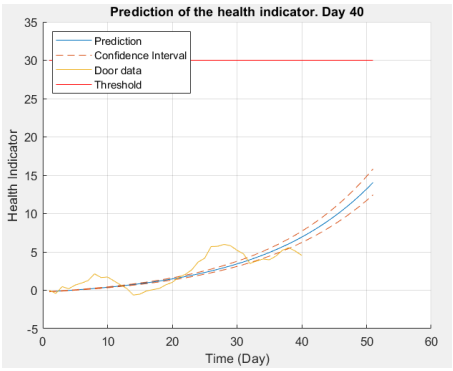
(a) RUL prediction on 10th day.



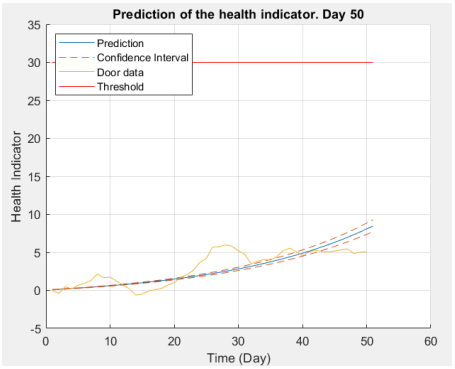
(b) RUL prediction on 20th day.



(c) RUL prediction on 30th day.



(d) RUL prediction on 40th day.



(e) RUL prediction on 50th day.

Figure 4.2: Predictions of the degradation algorithm.

The toolbox enables the generation of diagnostic plots for the prediction. Figure 4.3 shows the difference between the predicted RUL and the actual(expected) RUL in each day. It can be clearly seen that the prediction converges to the real value as the amount of data increase.

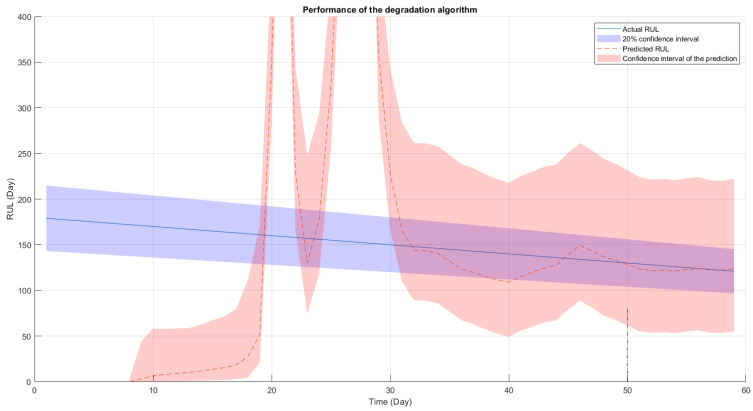
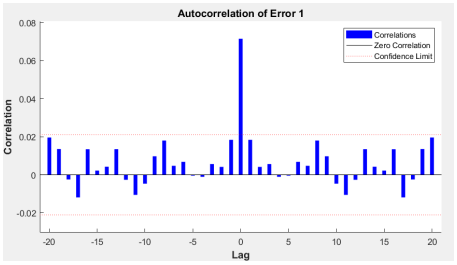


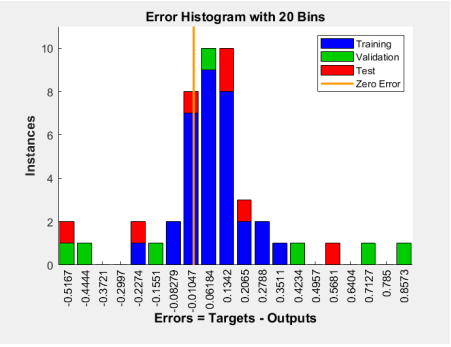
Figure 4.3: Comparison of predicted and real RUL.

4.1.3 Artificial neural network

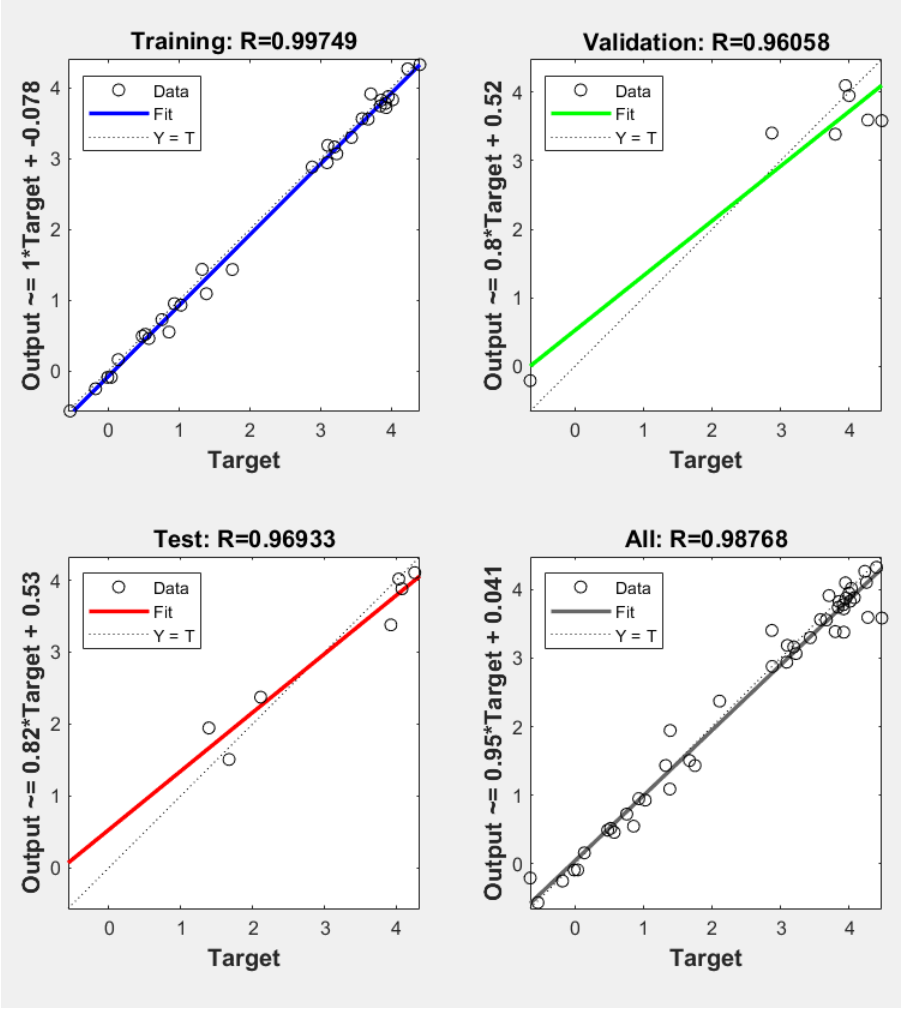
The network is trained with the first 50 days of the health indicator data and predictions for 10 days between 50th and 60th days were made to test the algorithm. Similar to the previous two methods, first 50 days are used as the training set to train the model, and the remaining 10 days are used as the test set to evaluate the performance of the network. In addition to that, to evaluate the performance of the network, the diagnostics plots that are given in Figure 4.4 can be used.



(a) Autocorrelation of errors.



(b) Histogram of errors.



(c) Regression plots for training, validation, test sets and overall fit.

Figure 4.4: Diagnostic plots of the neural network.

Figure 4.4a demonstrates the autocorrelation of the errors. It can be seen

from the figure that all of the errors except the one with zero lag are within the 95% confidence interval, meaning that the errors are not correlated with each other. In other words, it is white noise. This shows that the prediction model is applicable.

According to the error histogram in Figure 4.4b, the highest amount of errors are clustered near the zero error point. The rest of the errors show a Gaussian distribution, meaning that the error of the fit is acceptable.

The regression plots in Figure 4.4c explain how well the fit performs on the test, training and validation sets. The overall accuracy according to those plots is 98% while giving 99% accuracy on the training set. The lowest accuracy is on the validation set with 96% accuracy. The fit is adequately accurate.

The response of the fit is given in Figure 4.5. The predictions will base on this function.

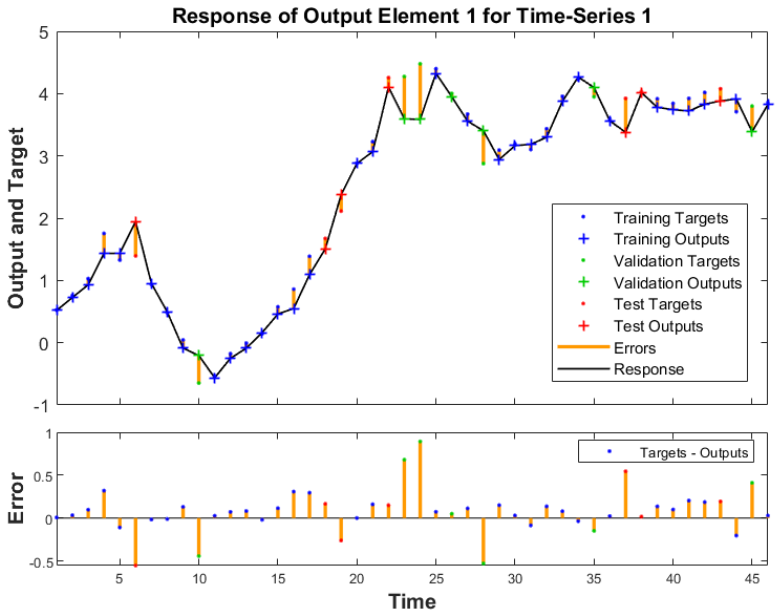


Figure 4.5: Response of the fit.

The neural network can clearly capture the pattern in the data and it is able to predict with high accuracy. However, even though the fit is excellent for training and test sets and short term predictions, the neural network can not predict the RUL since the time that needs to be predicted is approximately 200% of the available data while the neural network is programmed to predict for only 15% of the entire data set, meaning that only 10 days of accurate prediction can be made with this algorithm.

4.2 Comparison of the learning methods

The performances of the three algorithms are compared on their accuracy on the test set. Their predictions on the interval between 50th and 60th day should be as close to the actual RUL as possible.

First of all, the predictions on the training set are also a good indicator of the presence of degradation in the component. For the regression and degradation algorithms, the fit on the training set is represented with Table 4.1. The red cells in the table represent the period where no degradation detected in the data. The green cells are the points where a clear degradation is detected by the algorithm. From this table, it can be concluded that the left lower arm strain gauge is not degrading, meaning that the component is not showing any sign of failure. This would also mean that the predictions for that component may not have adequate accuracy. The frame strain gauges have a clear degradation starting from the 20th day. This would indicate that the first component that will fail is probably one of the frames. The prediction for those components would also have higher accuracy than the others due to the clear degradation.

	Right										Left										
	Frame		Acc X		Acc Y		Acc Z		Current		Frame		Underlevel		Acc X		Acc Y		Acc Z		RUL
	R	D	R	D	R	D	R	D	R	D	R	D	R	D	R	D	R	D	R	D	
1	.	.	.	.	.		.	.	.	.	.	.	.	.	.	.	.	.	.	.	179
2	.	0	.	.	.	0	.	.	.	.	.	.	.	.	.	.	.	.	.	.	178
3	0	35	7	13	.	4	10	30	.	2	.	1	.	78	.	88	.	102	.	72	177
4	0	70	.	5	.	9	95	19	.	2	.	1	206	26	.	29	.	36	12	111	176
5	1	202	.	.	.	15	.	.	.	.	.	1	.	6	.	52	.	35	.	132	175
6	.	135	.	.	.	19	.	.	.	.	.	1	175	4	.	43	.	51	.	153	174
7	2	.	.	25	.	22	23	24	.	6	7	1	.	3	2	40	.	85	94	206	173
8	4	1	.	30	.	0	33	30	.	5	1	.	.	.	.	.	.	164	25	.	172
9	6	4	.	37	.	0	108	27	.	7	4	.	.	.	1	.	.	384	154	179	171
10	5	5	.	98	.	20	105	36	.	8	.	3	.	8	.	69	.	460	.	124	170
11	5	6	.	138	.	305	230	57	.	7	.	6	.	19	11	117	.	225	.	.	169
12	6	7	.	131	.	0	489	234	.	7	4	8	.	38	29	139	.	209	2	.	168
13	8	9	.	126	.	190	318	218	.	7	4	9	.	57	64	141	.	.	5	64	167
14	12	11	.	125	.	137	386	166	.	6	11	10	174	71	.	138	.	.	8	101	166
15	13	14	.	125	.	135	651	145	.	6	14	12	170	81	.	128	.	55	9	128	165
16	15	17	.	125	.	136	.	139	.	6	10	14	221	89	.	127	.	137	21	157	164
17	20	22	.	128	.	142	.	139	.	9	14	15	317	95	.	132	.	127	95	225	163
18	22	28	.	134	.	153	.	144	.	17	12	18	572	120	.	140	.	123	.	420	162
19	168	48	.	143	.	168	.	151	.	32	4	27	234	.	.	152	.	125	.	.	161
20	31	62	29	158	.	192	.	165	.	58	4	51	279	.	3	171	.	134	1	.	160
21	33	62	.	185	.	137	.	186	.	218	7	354	78	163	5	205	.	148	2	269	159
22	33	73	.	149	.	156	21	217	.	133	.	386	.	227	8	291	.	173	3	155	158
23	33	78	.	.	.	.	30	271	80	126	92	230	99	200	10	.	.	204	4	108	157
24	34	78	.	.	.	415	39	427	.	131	100	130	22	215	.	.	.	334	4	88	156
25	35	75	.	365	.	209	44	.	.	132	112	177	70	.	.	239	.	.	.	73	155
26	37	76	.	228	.	152	51	.	.	136	110	324	110	244	2	159	.	.	6	65	154
27	39	77	.	179	.	120	.	300	.	140	109	.	140	202	3	117	.	449	8	61	153
28	43	80	.	152	.	106	.	210	14	.	114	.	181	185	5	98	.	420	10	60	152
29	46	87	.	132	.	99	.	165	.	.	59	.	.	176	10	87	.	352	16	63	151
30	48	92	.	117	.	94	.	135	.	.	130	361	.	.	12	82	.	376	20	67	150
31	49	98	.	107	.	92	68	117	.	.	144	228	.	.	13	79	.	437	24	72	149
32	51	99	.	100	.	89	73	104	.	141	166	165	.	.	14	79	.	469	31	79	148
33	51	97	.	96	.	89	111	97	.	158	.	143	465	.	10	83	.	467	40	87	147
34	53	95	.	94	.	89	157	93	.	145	155	143	.	.	14	86	.	.	21	96	146
35	55	91	.	92	.	90	183	92	.	139	177	140	.	.	16	89	.	.	17	100	145
36	57	89	51	90	.	92	136	92	.	153	353	132	.	.	18	92	.	.	23	103	144
37	60	88	56	89	.	95	157	91	.	152	164	123	.	.	22	94	.	317	24	105	143
38	63	88	62	90	5	99	379	89	.	153	182	120	.	.	26	95	.	229	54	106	142
39	67	88	64	91	17	102	651	89	.	152	99	115	.	.	32	96	.	186	215	108	141
40	.	90	70	92	3	106	.	88	.	194	121	116	.	.	38	98	.	163	223	109	140
41	78	94	75	93	.	108	.	87	.	287	178	109	.	.	46	102	.	153	232	112	139
42	80	98	84	96	27	112	.	87	.	.	177	115	.	.	54	104	20	147	.	116	138
43	88	102	95	98	3	117	.	89	.	192	207	120	.	.	64	106	8	145	248	119	137
44	93	108	99	101	133	123	.	93	.	152	128	125	.	.	83	109	20	142	299	123	136
45	99	114	102	104	.	126	.	96	.	163	120	128	.	.	107	113	.	139	307	126	135
46	105	119	107	108	.	128	.	101	.	159	122	129	.	.	118	116	.	136	336	129	134
47	111	122	106	112	.	129	.	107	.	147	.	149	.	.	102	119	40	133	233	130	133
48	117	126	110	117	.	129	.	114	.	138	218	145	.	.	110	121	.	130	.	130	132
49	123	128	123	123	.	130	276	123	26	132	127	127	.	.	118	126	90	130	118	131	131
50	130	130	130	130	132	130	130	130	120	130	130	132	130	.	127	130	128	130	130	130	130

Table 4.1: Results on the train set for regression and degradation algorithms.

On the other hand, the performances of the algorithms on the test set are stated in Table 4.2. The red cells indicate predictions with higher than 20% error. The rest are represented with a colour scale, with lighter colours indicating higher accuracy. As can be clearly seen, Neural Network performs superior to the other two algorithms. The degradation function of MATLAB's PdM toolbox also captures the pattern with considerable accuracy for some of the

sensors. However, the basic exponential regression can not perform an accurate prediction except the right frame strain gauge.

	Right															Left												
	Frame			Acc X			Acc Y			Acc Z			Current			Frame			Underlever			Acc X			Acc Y			Acc Z			RUL
	R	D	NN	R	D	NN	R	D	NN	R	D	NN	R	D	NN	R	D	NN	R	D	NN	R	D	NN	R	D	NN	R	D	NN	
50	130	130	0,00%	130	130	0,00%	132	130	0,00%	130	130	0,00%	120	130	0,00%	130	132	0,00%	130	.	0,00%	127	130	0,00%	128	130	0,00%	130	130	0,00%	130
51	140	132	4,20%	138	137	6,00%	184	130	3,53%	.	136	15,41%	.	110	7,85%	132	128	0,23%	149	.	7,70%	137	133	11,50%	.	132	129,00%	.	130	5,70%	129
52	145	133	1,30%	157	144	9,20%	217	133	5,79%	.	144	31,91%	.	110	8,51%	134	123	0,73%	222	.	15,90%	147	138	18,70%	91	135	42,00%	224	130	1,57%	128
53	151	136	1,60%	165	151	1,46%	.	135	7,60%	.	154	22,89%	.	114	0,87%	134	121	0,16%	341	153	16,12%	159	142	12,50%	121	139	74,00%	276	132	0,98%	127
54	157	138	3,90%	173	166	2,23%	.	137	3,73%	.	163	9,63%	.	125	8,21%	139	122	0,70%	.	179	15,34%	170	147	0,04%	.	142	20,00%	.	133	3,27%	126
55	163	141	4,60%	190	163	9,98%	.	140	4,56%	.	172	19,87%	.	158	10,89%	147	121	0,20%	.	243	0,90%	192	130	2,80%	.	144	25,00%	236	135	0,44%	125
56	169	143	6,00%	224	176	19,50%	.	145	10,66%	205	186	19,14%	.	253	25,60%	144	122	0,06%	533	317	10,40%	194	153	0,70%	366	147	2,99%	289	137	7,33%	124
57	175	145	2,60%	254	185	23,90%	.	150	4,62%	218	204	6,53%	.	486	47,38%	148	123	0,39%	.	399	7,30%	207	157	6,90%	.	150	4,28%	.	139	3,54%	123
58	182	148	0,70%	306	196	9,55%	315	155	6,88%	278	219	17,90%	17	509	47,47%	569	122	0,37%	.	466	4,60%	219	161	12,80%	.	153	2,96%	292	141	6,88%	122
59	185	151	0,58%	.	205	6,80%	179	164	6,80%	216	235	9,70%	.	.	20,38%	570	122	0,15%	.	493	2,75%	271	163	14,20%	.	155	7,75%	295	143	6,53%	121
60	195	153	0,22%	477	215	2,29%	119	170	13,11%	299	247	3,95%	.	.	25,34%	.	123	0,51%	.	433	0,21%	291	166	10,80%	.	159	25,41%	301	147	0,57%	120

Table 4.2: Results on the test set for all three algorithms.

4.3 Correlations of the parameters

One of the most important outcomes of the project is to observe and understand the effects of the parameters. The object of this study was to find out whether it is possible to see the trend in one parameter by investigating other parameters. This inspection step is crucial for the implementation of the predictive maintenance system since it is not feasible to mount 12 sensors to every bus door. In that sense, the relations between the signals obtained from different sensors are examined.

Comparing two signals directly is meaningless since the values that are measured have different characteristics. It is not possible to compare acceleration to current directly. Instead, the statistical features of those signals can be used to detect correlations. The features that were mentioned in Section 3.1.1 were used to compare two signals.

The goal is to remove the strain gauges in the implementation phase. The correlations between the statistical features of the strain gauge signals and accelerometers are investigated in that purpose. All of the heat maps regarding those correlations are given in Appendix A The correlations visualized with those plots are lower door arm - accelerometer (X), lower door arm - accelerometer (Y), lower door arm - accelerometer (Z), door mechanism frame - accelerometer (X), door mechanism frame - accelerometer (Y), door mechanism frame - accelerometer (Z), lower door arm - current and door mechanism frame - current.

The heat maps contain the three correlation matrices that were mentioned before. The top left square matrix of the heat map represents the self-correlation of the signal from the first sensor and the bottom-right square matrix represents the self-correlation of the signal from the second sensor. The two square matrices on the counter diagonal represent the correlations of the two signals and those two matrices are symmetric about the origin. The matrices on the counter diagonal can be investigated to explain the correlations between two signals.

As an example of this, Figure 4.6 represents the correlations between the left frame strain gauge and left accelerometer(X). It can be seen from the top-right part of the heat map that there are 75% correlations in some of the features, but not a significant relation. In order to replace the strain gauges with accelerometers in the data analysis part, clear information on those correlations must be captured. Field tests have great potential to identify those relations more clearly.

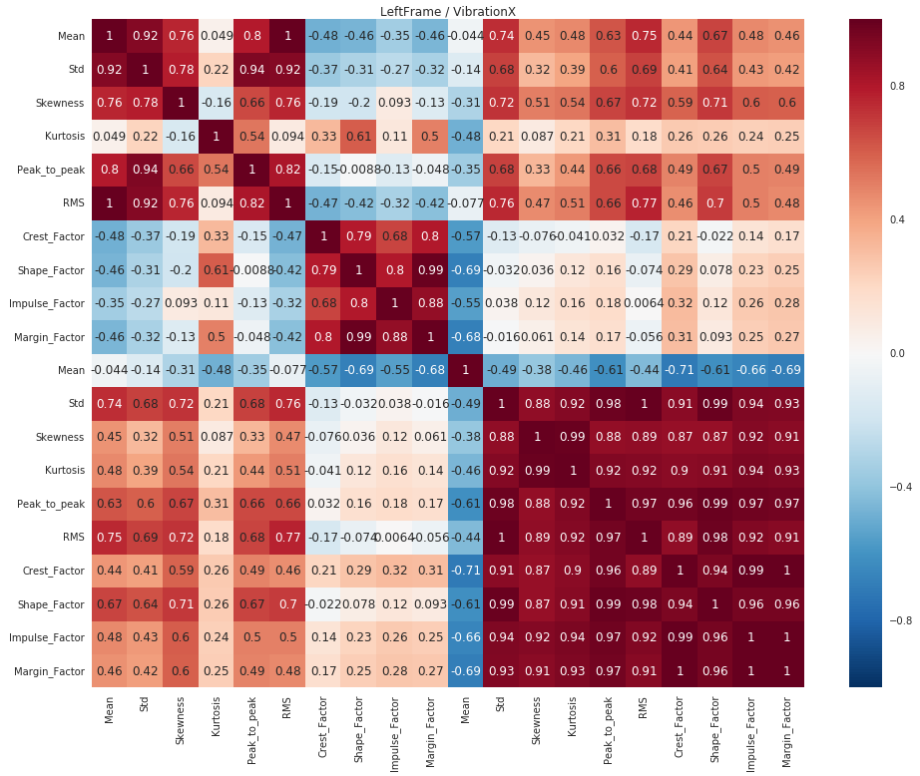


Figure 4.6: Correlation matrix of the frame strain gauge - accelerometer x axis.

From the results obtained from this analysis, it is not possible to eliminate any of the strain gauges with the accelerometers. There are not any significant correlations between parameters. Additional sensors or a new placement diagram must be used to detect more significant interactions.

4.4 Cost analysis

The main costs of the project depend on 3 factors: the CM equipment, the data storage and analysis software and the test sets in the company . The estimated costs for the elements that were used in the project and the total estimated cost of the project is summarized in Table 4.3.

Expense type	Unit cost	Number	Total Cost
Sensors			
Strain gauge	12 €	6	72 €
Accelerometer	400 €	2	800 €
Temperature sensor	78 €	2	156 €
Shunt resistor	82 €	1	82 €
Data collection setup	4500 €	1	4500 €
Software			
SensorCloud License	150 $/mnt	1	150$
MathEngine Extension	50 $/mnt	1	50$
MATLAB	800 €/yr	Optional	800 €
MATLAB PdM Toolbox	680 €/yr	Optional	680 €
Factory test			
Test setup	1500 €	1	1500 €
Total annual expenses			200 € (+1480)
Total non-recurring expenses			7110 €

Table 4.3: Estimated expenses for the project.

The software does not cause a great financial expense on the project thus a more advanced solutions can be used for better performance. However, still a more cost efficient algorithm can be found after further research. The data collection setup creates more than 50% of the total expenses, meaning that if an alternative solution is available, that would decrease the general costs greatly.

5. Conclusions

The project aimed to inspect the current progressions in predictive maintenance systems and adapt it to the bus door manufacturer, Ventura Systems according to its goals and expectations. The plan of approach consisted of three main steps;

1. Investigation of case studies and relevant progression about PdM to prepare a convenient methodology for the project in the company.
2. Constructing the structure of the project based on similar studies. Planning the project steps. Preparation and design of the experiment.
3. Analysing the results and evaluating the applicability of the model in the real doors. Performance comparison of several algorithms.

The study covered these three steps to achieve the main objective. The actions taken in those steps are briefly summarized.

1. **Research**
   Most of the information about maintenance systems is provided by programming suppliers. In this case, MATLAB published several comprehensive guidelines and example projects to aid engineers working in PdM projects. In addition, many case studies around the world investigate algorithms and evaluated the implementation phase of their project. These studies provide a head start for organisations that plan to adopt the new methods of maintenance. Having a vision of what is needed and what might cause problems enhanced the progression speed and robustness of the study greatly. The methodology was based on the knowledge from case studies as well as the possibilities in the company.
2. **Planning**
   The methodology was planned to be a recurring process to have the deepest insight about the door and mechanism failures. Initially, an experimental test set was used to determine the most influential parameters and to decide if additional information is necessary or not. The experiment data was analysed to extract the desired information. Based on that information, an extensive endurance monitoring test is designed.
3. **Application**
   The data collected from the endurance test is analysed with several different methods to ensure the reliability of the study and to evaluate the performances of different algorithms. Three tools used in this step are exponential regression based on least-squares curve fitting, exponential degradation based on probability distribution and artificial neural network. The performances of these three methods are compared based on their accuracy to estimate the actual remaining useful life.

The successful implementation of the three design steps enabled the solution to the main goal and interest of the project:

"Designing a predictive maintenance algorithm for bus door systems."

The basic algorithm did not give satisfactory results in the study but it can easily be improved with larger data for commercial use. More advanced algorithms increase the performance but also cause greater expenses for the company and require more computational power. The oncoming steps of the project are the recurrent steps that were mentioned for performance and precision improvement.

5.1 Limitations of the study

The project has limitations in several areas, naturally. The data used in the learning model is not comprehensive enough to design a fully operable PdM system. The data contains limited information from a single test set, which enables the algorithm to make simple predictions about whether it is about to fail or not, rather than informing about the type of the failure that may occur. Also, a single test set means that the validation of the data set is not yet done. The number of tests must be increased to ensure that the data collected from the test is not biased. Therefore, the algorithms used in this project are in the simplest way and aim to provide a reliable result but not to optimize accuracy. The findings of the analyses would be able to estimate the desired information but not provide sufficient performance if they are implemented in a real system directly.

Another limitation is about the quality of the data. Conducting a test in a test environment is different from testing it on a field. Testing the same parameters on a real bus may give similar, but not the exact same results. As an example, eigenfrequencies of the door may shift slightly due to elasticity of the connection parts on the bus or due to the vibrations of the bus. That effect must be observed and removed from the signal before using the algorithm for improved precision. Also, it is not possible to measure all the values used in the test on a real bus. For instance, implementing strain gauges can be a problem in the field test. Hence it is vital to correlate the parameters from strain gauges to other types of sensor measurements. A similar operation has been done in Figure 3.5 of Section 3.1.2 to find the correlation between parameters and remove the redundant ones. However, as stated before, this study does not represent all the possible failures in a door and a redundant parameter for a failure type can be necessary for another one. These relations must be observed in field tests before making robust conclusions and implementations on a real bus.

5.2 Recommendations for the company

The report is a simplified guideline to build an extensive predictive maintenance system. The methodology used in the project can be generalised to design a fully comprehensive algorithm. However, there are significant points that need to be approached with caution while generalizing it.

- Most important of those is scaling step by step to avoid undesired errors. The next objective is to identify the failure types and associate them with

the features extracted from the data. A correlation study has been done in this project, however, for different measurements, different parameters can have particular significance. Those relations must be studied carefully to connect the indicators to specific failures.

- Another important factor is identifying correlations from parameters for each failure type. The limitations in the field test obligate the inspection of correlations. If the parameter is not directly connected to another parameter, the correlations in some of the features or combinations of several features from one or many data sets may be used. However, to obtain this knowledge largely scaled tests must be done beforehand. An efficient field test set can only be implemented after that.
- Next problem that the company may face is the condition differences between the factory and the field. The sensors in the bus, especially the accelerometers will measure the noise caused by the movements of the bus. Several methods can be used to eliminate this noise in accelerometer readings.

Removing time-domain features
This is the simplest and most straightforward solution. The vibration of the bus changes time-dependent measurements (i.e. mean) but not the features in the frequency domain(i.e. eigenfrequencies). Spectral features of the door can easily be isolated from the vibration signals of the bus, thus, there will not be any need for extra computations or equipment.
However, this prevents the usage of time-domain features. Peak to peak amplitudes, mean, distribution of the signal will be affected by the bus vibrations. This solution limits the analysis to spectral features and it may cause loss of precision of any time-domain feature is significant in determining the health indicator of the door.
Also, spectral features of the door must be well understood to distinguish from the noise coming from the bus.
Pros
Easy, no need for new sensors or equipment provides sufficient information on vibration.
Cons
Losing the ability to work with time-domain features, losing the correlation to other parameters through the time-domain features.

Measuring and subtracting the bus vibration signals
Another straightforward solution is directly removing the bus vibration signal from the door vibration signal. Theoretically, it is applicable. however, in practice, it needs perfect precision. The signal removal operation must be done for exactly the same data points. Small deviations may cause higher frequencies to be less informative.
This is a simple method but can give the desired accuracy if could be applied properly. The problem is it does not allow any time-based error on measurement. A slight offset can cause large errors, especially on a higher frequency. Pros
Gives an accurate result. Relatively simple algorithm.
Cons

Hard to implement, needs the 3rd sensor, 0 error margin.

FEM modelling
Looking through the properties of the door and the bus separately in a FEM software can be used to identify the features of both signals. For example, if the door has eigenfrequencies at 90 and 120 Hz according to FEM software and the bus has 15 and 60 and if the accelerometer gives eigenfrequencies of 13, 65, 80 and 105 in the field experiment, it can be confirmed that 13 and 65 belong to the bus while 80 and 105 belong to the door. The values from the bus can be removed by comparing with the FEM software and continue the calculations with the doors.
Pros
Can be used both with the 3rd sensor or without it. Simple approach. Can work well with simple systems with low numbers of components.
Cons
Needs an accurate model of the door. Can be too complex to use in a FEM software. FEM software does not give 100% precision so results may not be enough to eliminate the signal from the door.

5.3 Future research

PdM methods require large data sets over long periods to perform in the most efficient operation. The algorithm that was designed in this project was based on several sensor data, but without the information of the distinct failure types therefore it can also be a base for a more comprehensive study. To enable the usage of the PdM algorithm effectively, the company needs to obtain more data, from both tests in the company and field measurements in real busses. Investigating several different failure types such as bolt and screw, engine, door arm and frame failures can provide better information about the features that change over time for those particular failures. Those features can be used to predict the type of failure in addition to the failure date, which would enhance the efficiency of the maintenance process greatly.

Another point of improvement can be using a different model in the design phase of the algorithm. The available data enables the usage of the degradation model for failure prediction as stated in Section 2.1. However, with full life-time data from several similar machines, it is also possible to build a similarity model, which would enable the improvement of the prediction. Collecting this kind of data can take much longer time and designing a prediction model based on that data can be even more complicated. That kind of project could be done in several years but in return, can greatly decrease the costs of maintenance operations and increase the efficiency by increasing the uptime for the doors.

Different algorithms can also be used in a single prediction model. In section 2.4, several studies that use various algorithms were mentioned. Exponential decay, neural networks, ARIMA models and AI are examples of the alternative algorithms. Exponential decay functions are used to predict the failure in this project. However, other methods of prediction can also give decent results and

can even be more accurate. Investigating the efficiency of the alternative algorithms can be the subject of a posterior study.

Interesting progress for the company could be applying a similar study for different types of doors. The study was conducted on a PSD. However, the 'Smart Machines' project is planned to be installed on every product of the company. ISD's, RSD's and OSD's have different response due to their mechanical and electrical differences. This indicates that each type of door may require a different algorithm and each algorithm may provide a broader insight into the doors. For instance ISD's exhibit larger vibration amplitudes during opening cycles than the PSD, which was the main focal point of this study. Investigating these kinds of differences can enhance the design process as well as the PdM systems within the company.

Acronyms

AI artificial intelligence. 14, 54

ANN artificial neural network. 35

ARIMA auto regressive integrated moving average. 12, 54

ARMA auto regressive moving average. 12

CBM condition-based maintenance. 1, 2, 15, 19

CM condition monitoring. 1, 4, 18, 19, 49

FFT fast Fourier transform. 22, 25, 36

IoT internet of things. 17, 18

ISD inward swinging door. 5, 55

OSD outward swinging door. 5, 55

PCA principal component analysis. 16, 24, 28

PDF probability density function. 40

PdM predictive maintenance. 2–4, 7, 8, 10, 11, 14, 16–20, 47, 51, 52, 54, 55

PM preventive maintenance. 1–4

PSD plug sliding door. 5, 30, 55

RM reactive maintenance. 1, 3, 4

RSD rapid sliding door. 5, 55

RUL remaining useful life. 11, 12, 14, 15, 18, 24–28, 34, 39, 41, 42, 45, 46

A. Correlation matrices

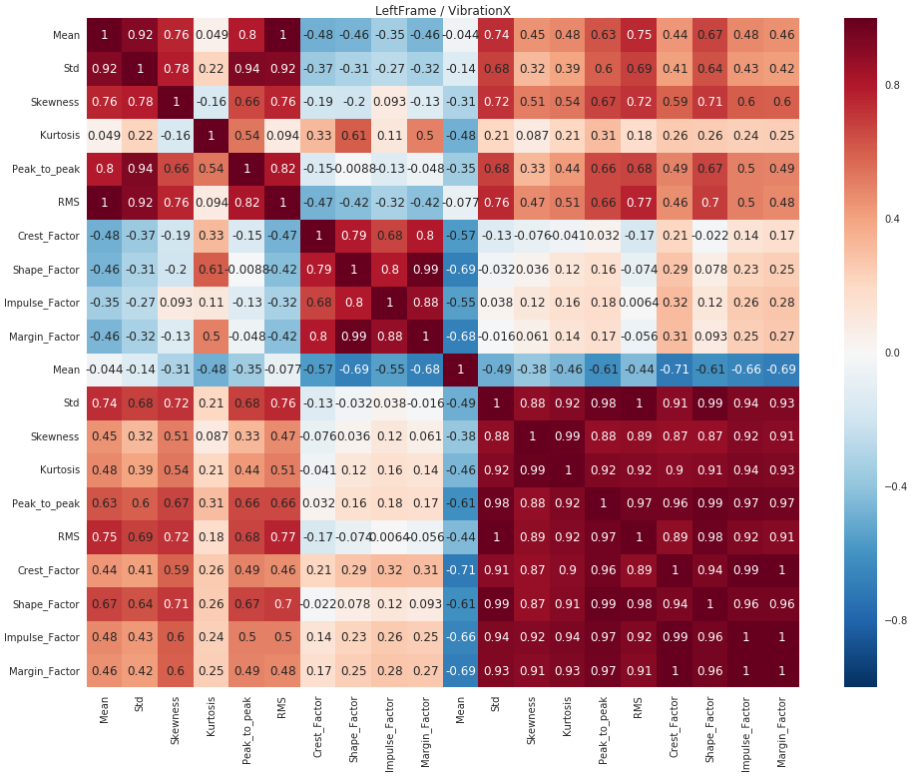


Figure A.1: Correlation matrix of the frame strain gauge - accelerometer x axis.

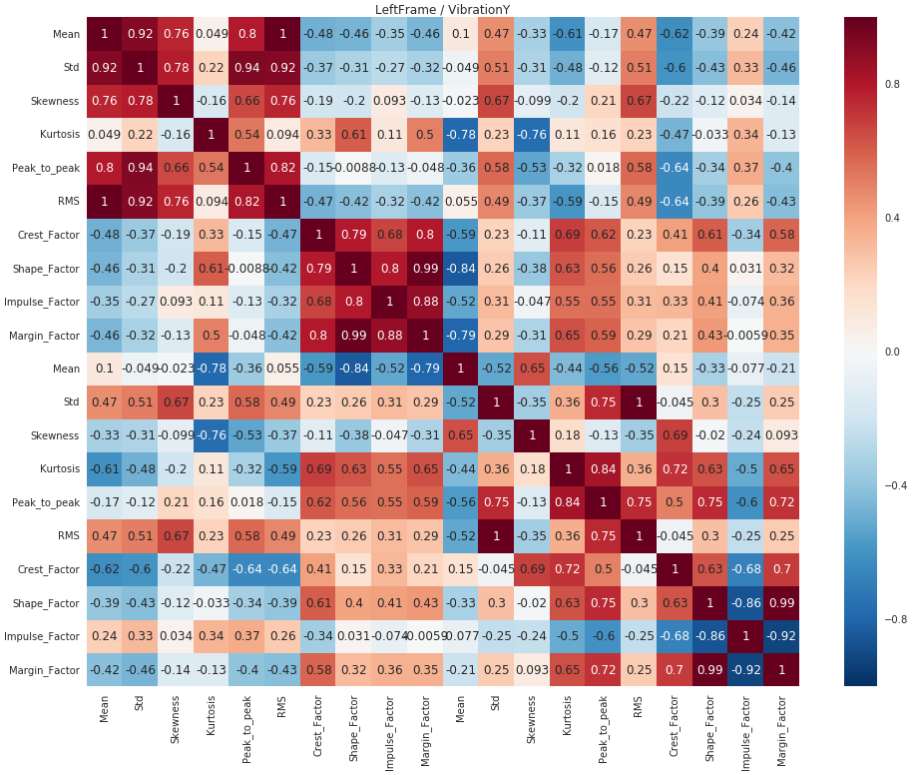


Figure A.2: Correlation matrix of the frame strain gauge - accelerometer y axis.

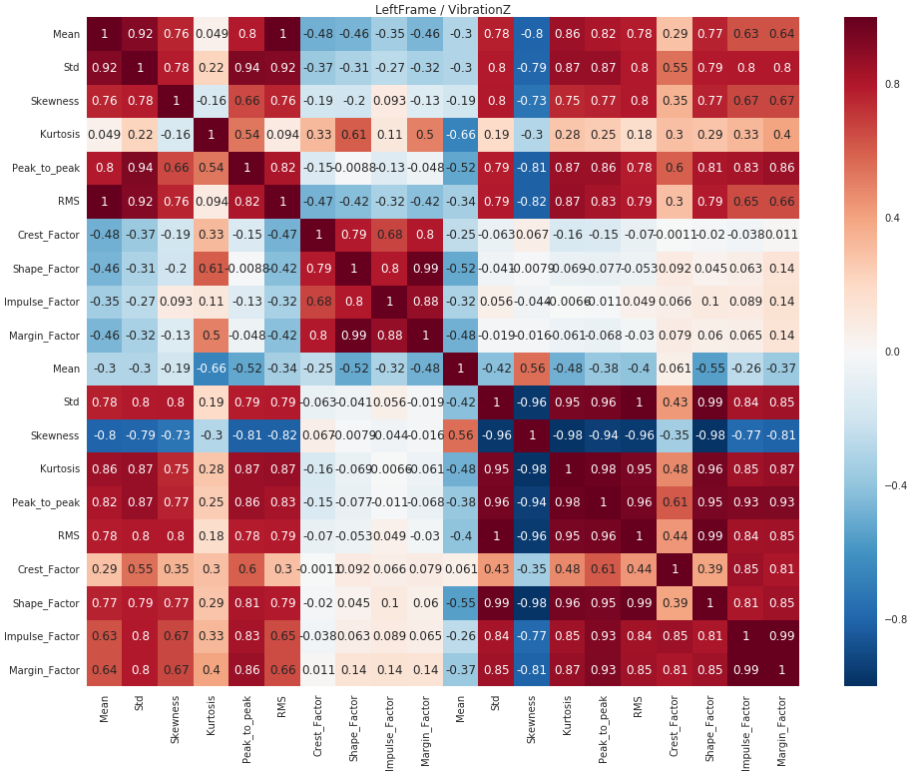


Figure A.3: Correlation matrix of the frame strain gauge - accelerometer z axis.

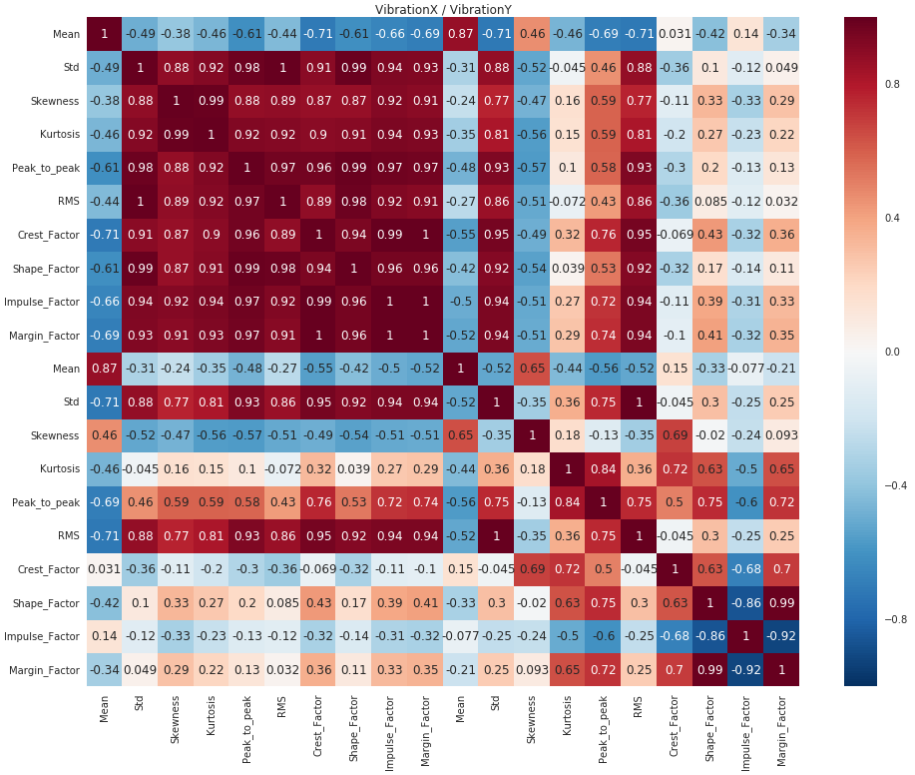


Figure A.4: Correlation matrix of the accelerometer x axis - accelerometer y axis.

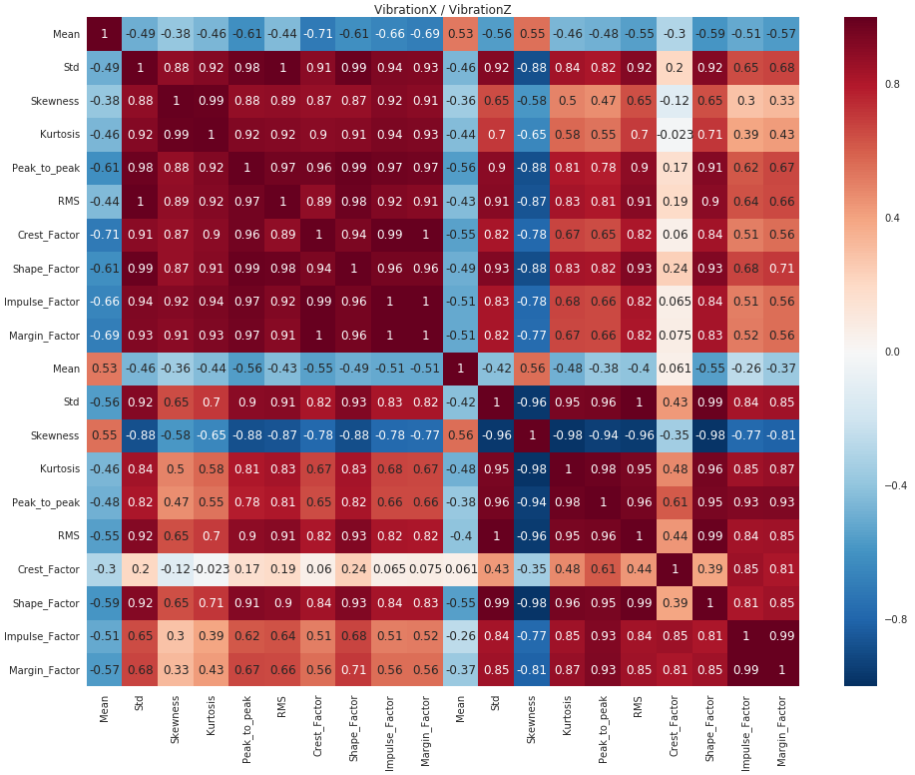


Figure A.5: Correlation matrix of the accelerometer x axis - accelerometer z axis.

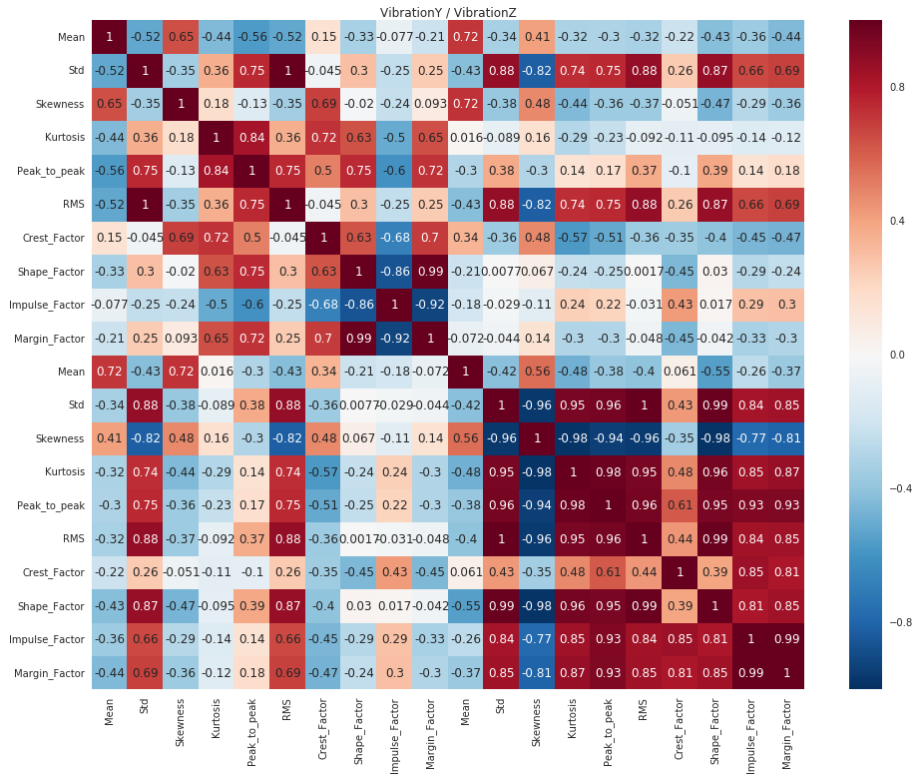


Figure A.6: Correlation matrix of the accelerometer y axis - accelerometer z axis.

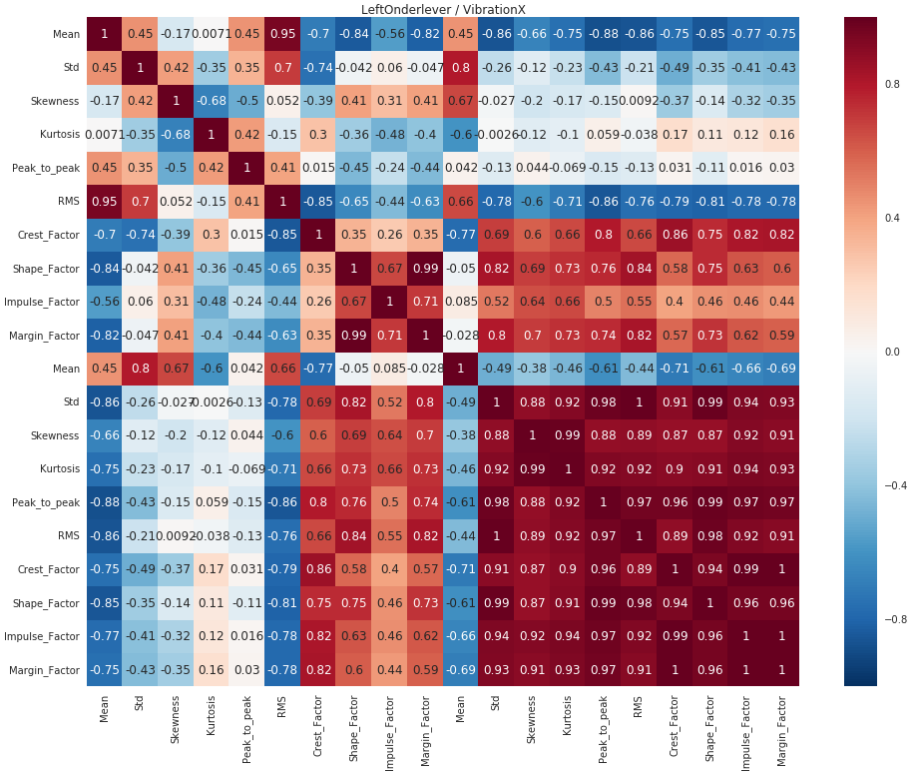


Figure A.7: Correlation matrix of the frame strain gauge - accelerometer x axis.

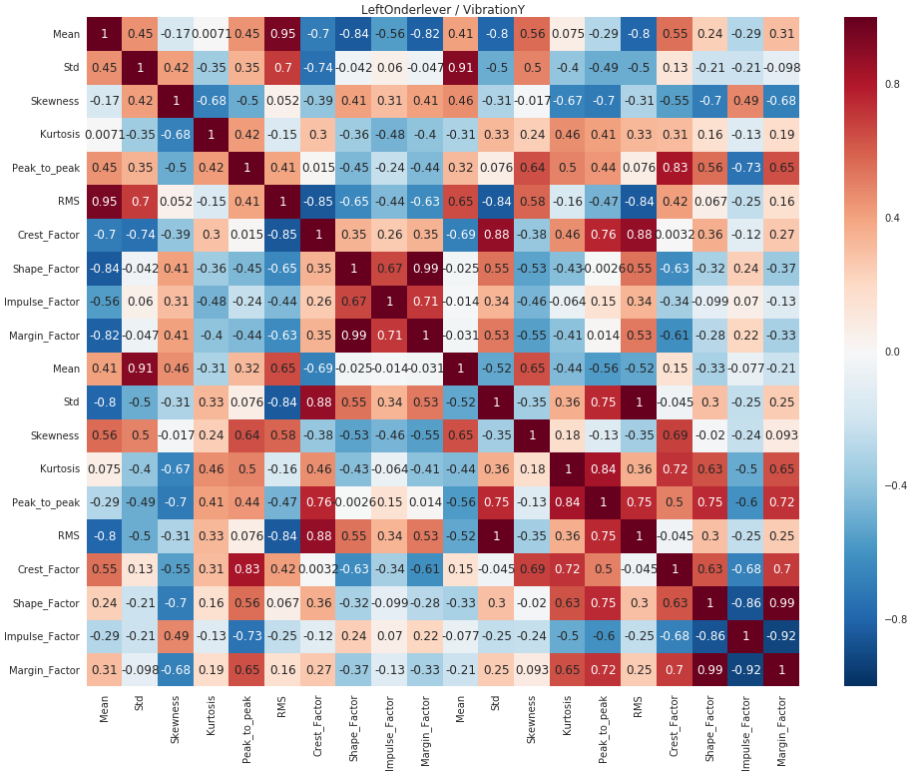


Figure A.8: Correlation matrix of the underlever strain gauge - accelerometer y axis.

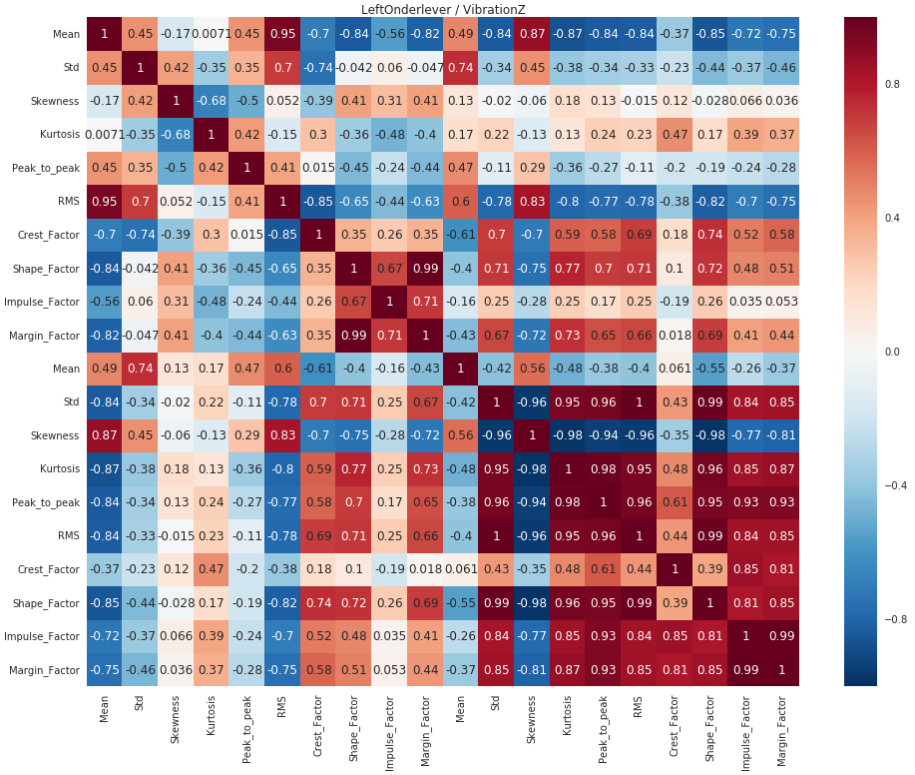


Figure A.9: Correlation matrix of the underlever strain gauge - accelerometer z axis.

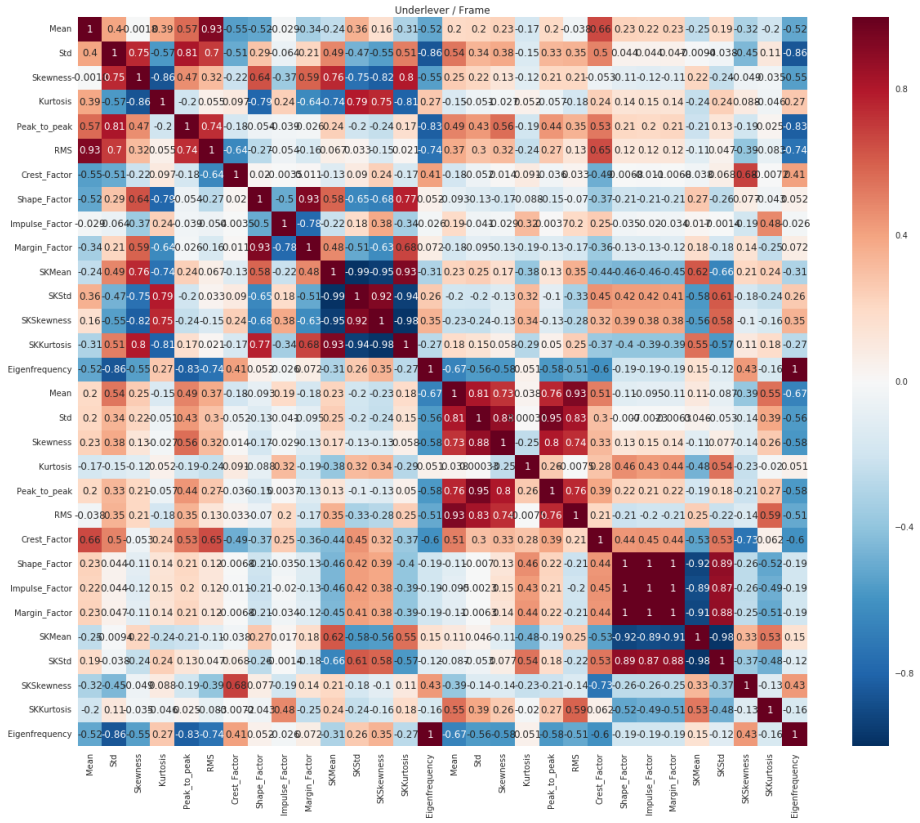


Figure A.10: Correlation matrix of the underlever strain gauge - frame strain gauge.

B. Predictions

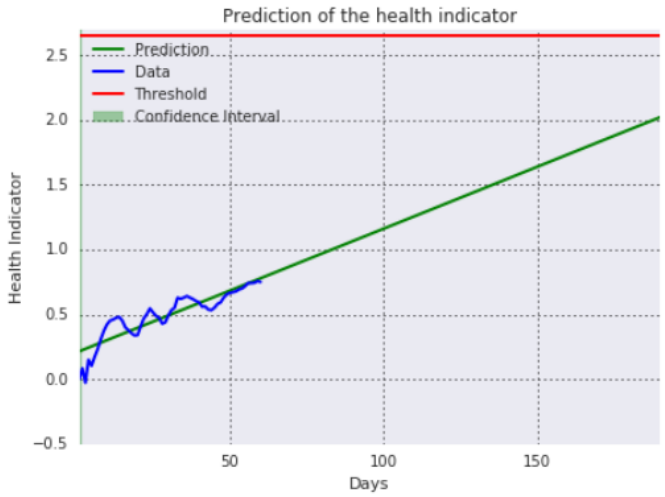


Figure B.1: Right frame condition prediction with regression model.

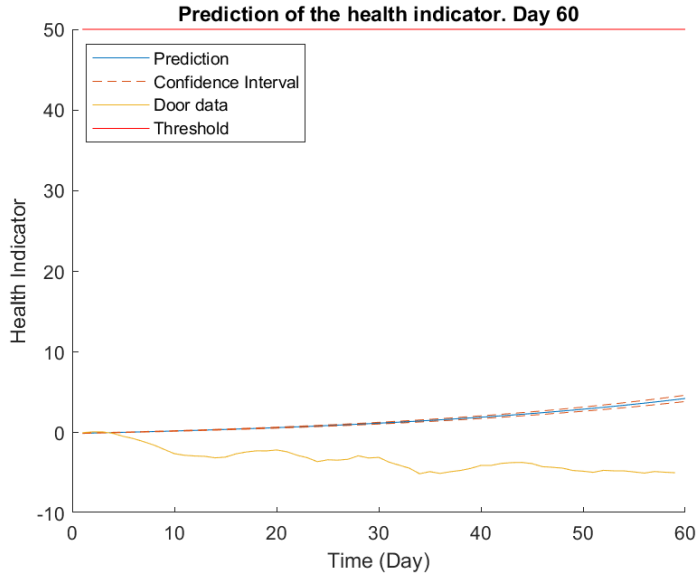


Figure B.2: Right frame condition prediction with degradation model.

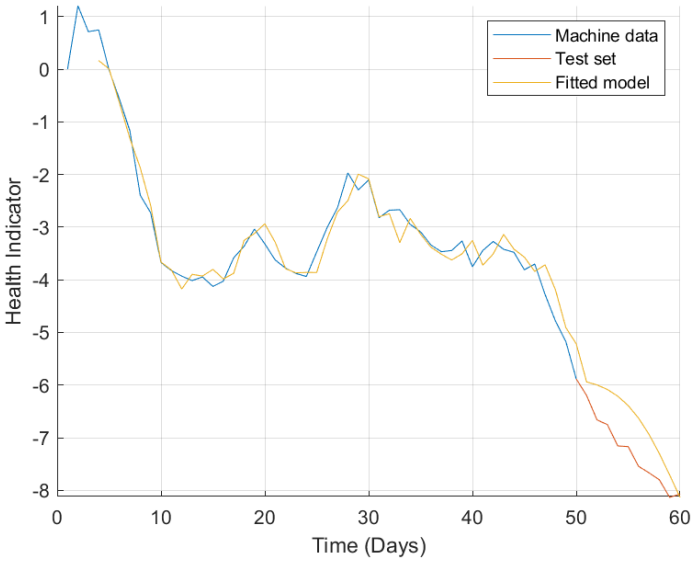


Figure B.3: Right frame condition prediction with neural network.

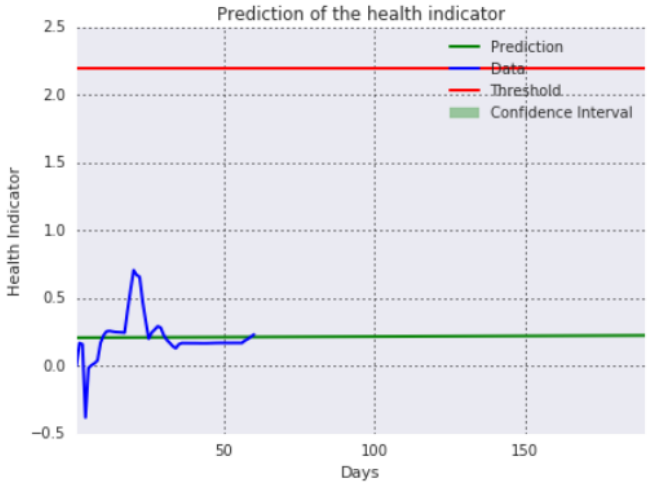


Figure B.4: Left frame condition prediction with regression model.

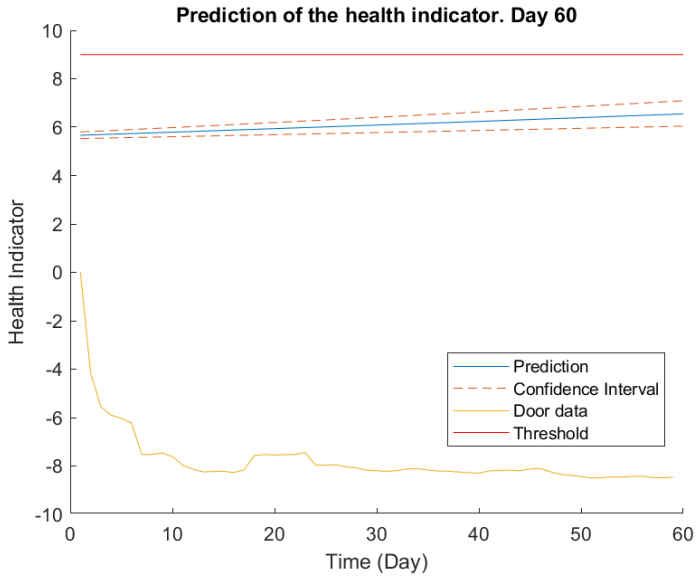


Figure B.5: Left frame condition prediction with degradation model.

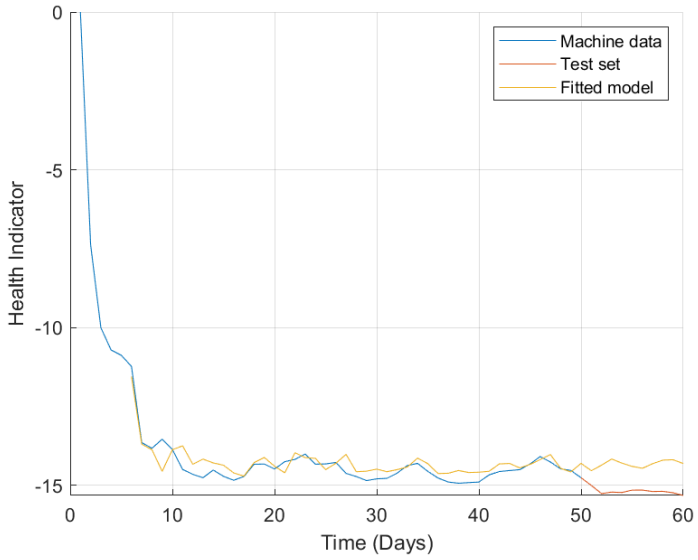


Figure B.6: Left frame condition prediction with neural network.

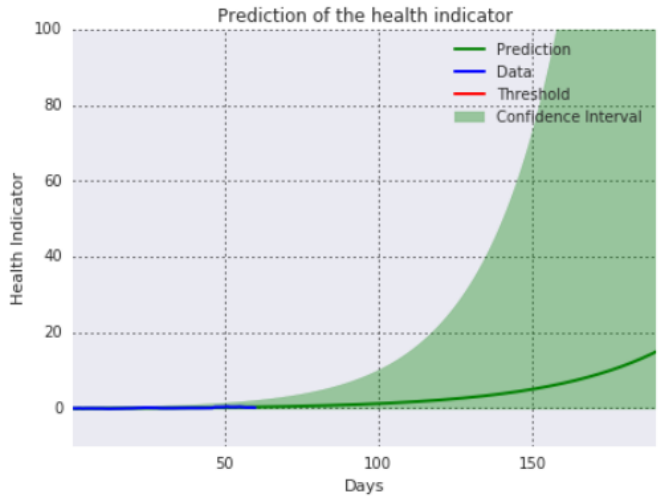


Figure B.7: Left underlever condition prediction with regression model.

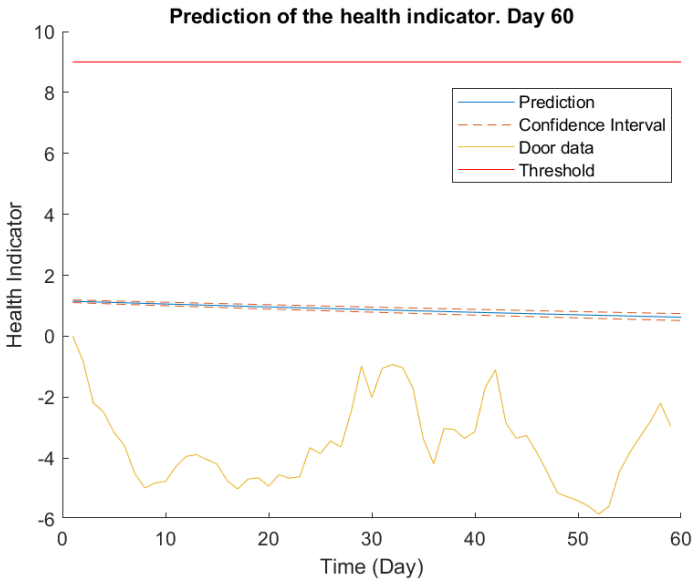


Figure B.8: Left underlever condition prediction with degradation model.

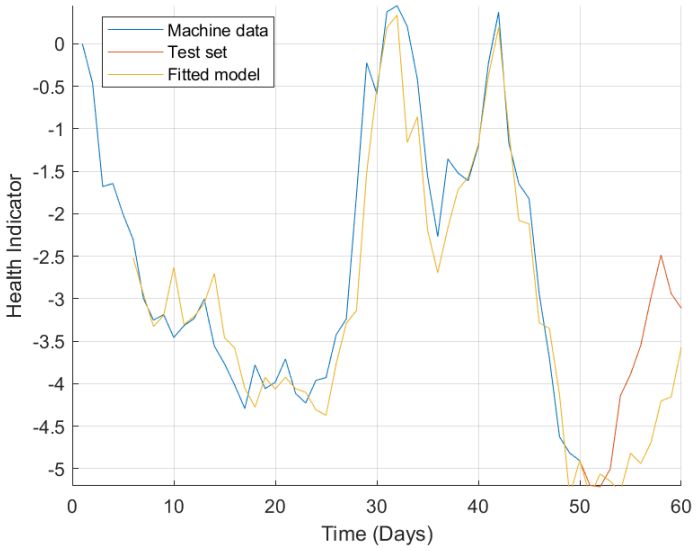


Figure B.9: Left underlever condition prediction with neural network.

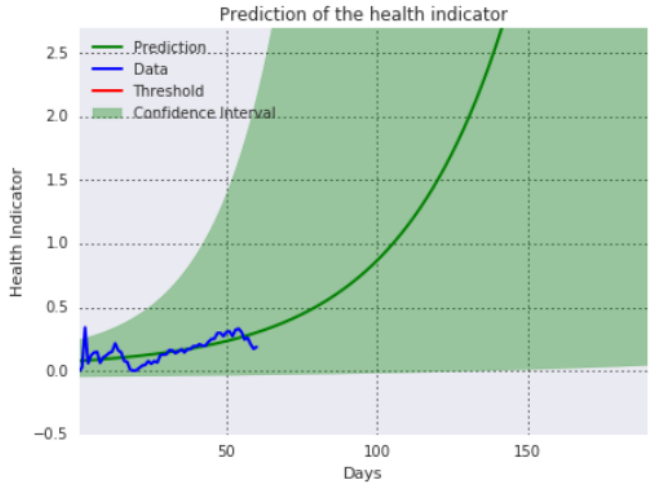


Figure B.10: Right accelerometer (X) condition prediction with regression model.

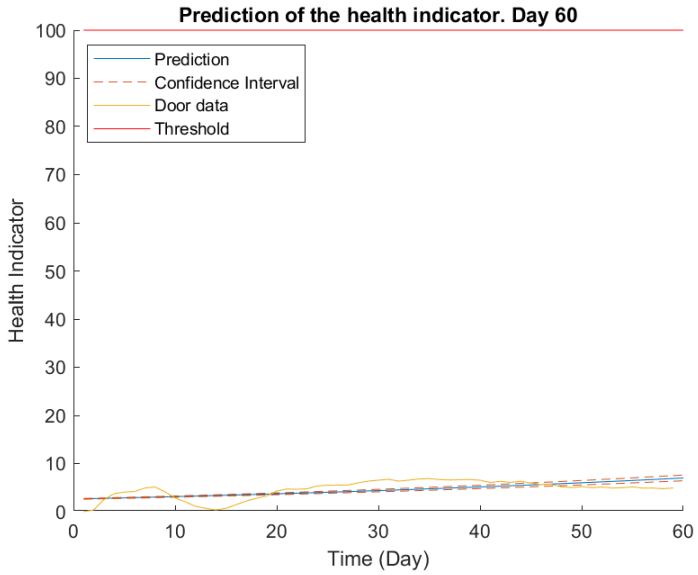


Figure B.11: Right accelerometer (X) condition prediction with degradation model.

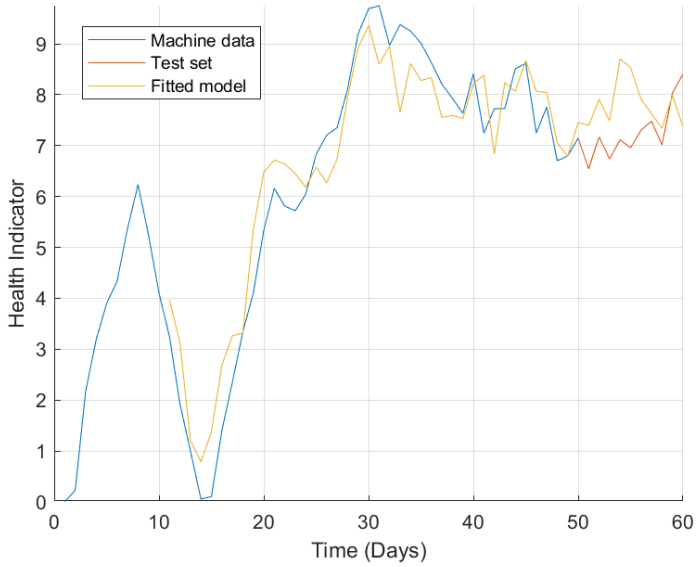


Figure B.12: Right accelerometer (X) condition prediction with neural network.

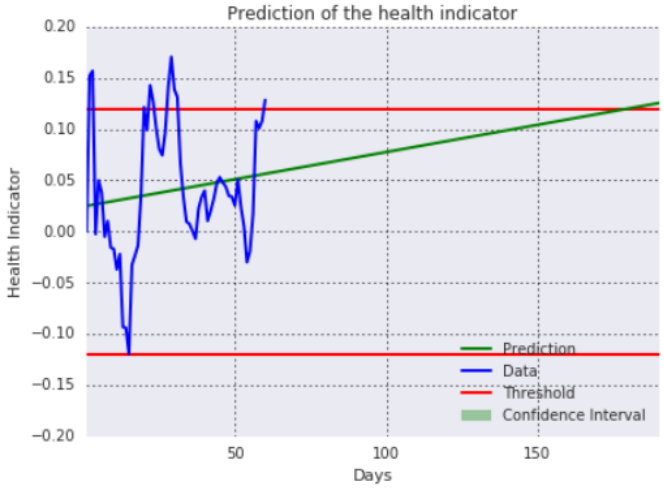


Figure B.13: Right accelerometer (Y) condition prediction with regression model.

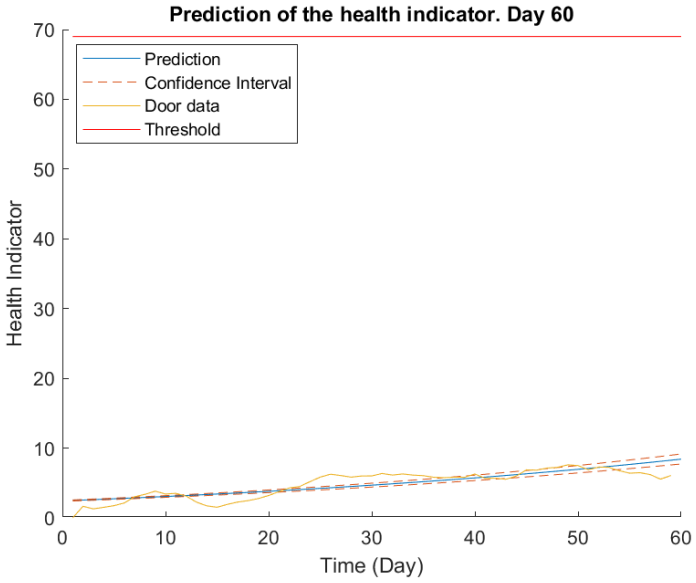


Figure B.14: Right accelerometer (Y) condition prediction with degradation model.

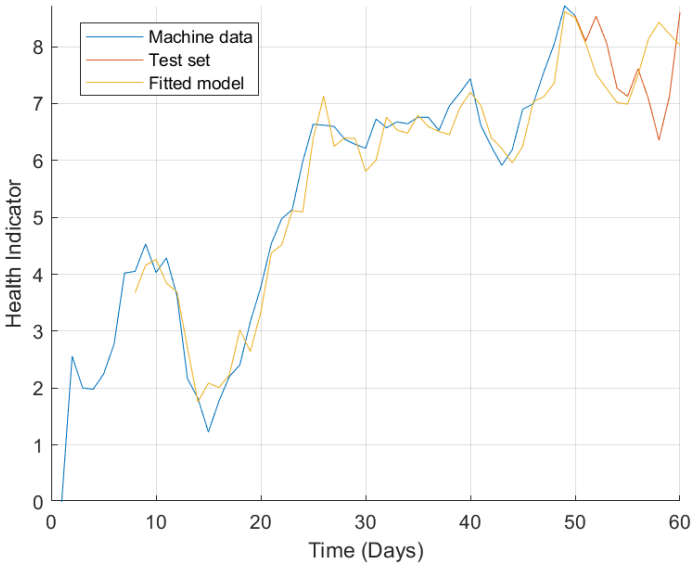


Figure B.15: Right accelerometer (Y) condition prediction with neural network.

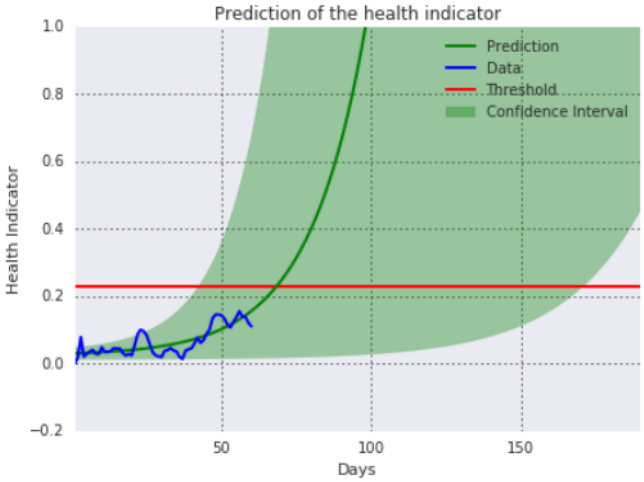


Figure B.16: Right accelerometer (Z) condition prediction with regression model.

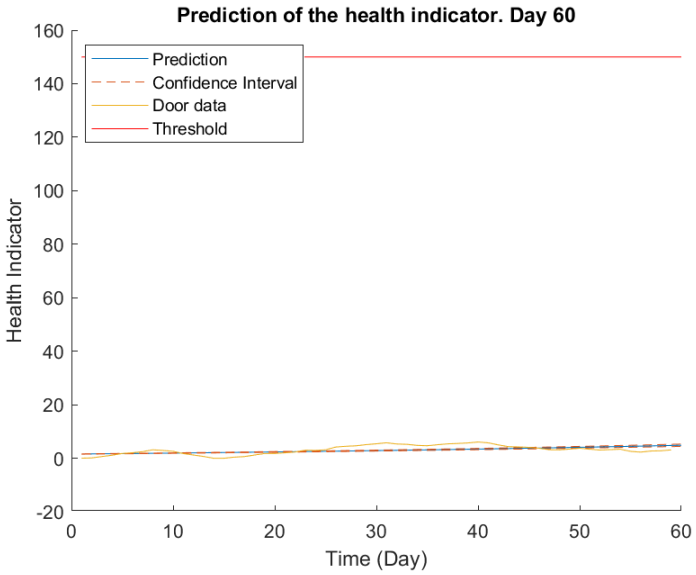


Figure B.17: Right accelerometer (Z) condition prediction with degradation model.

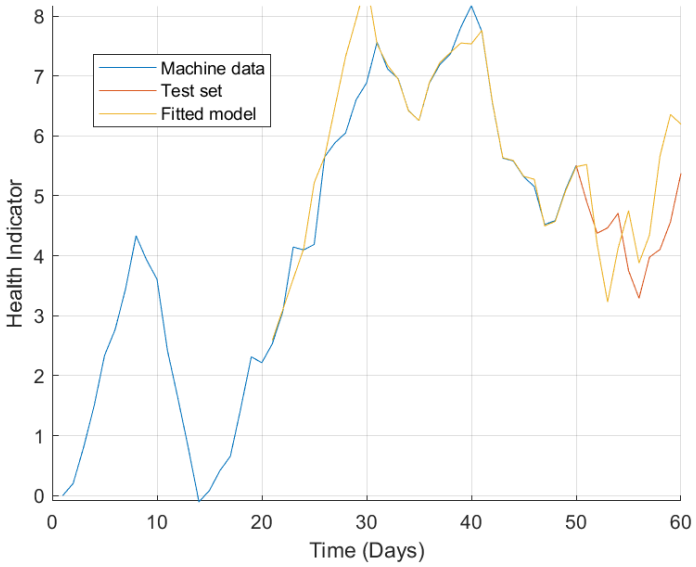


Figure B.18: Right accelerometer (Z) condition prediction with neural network.

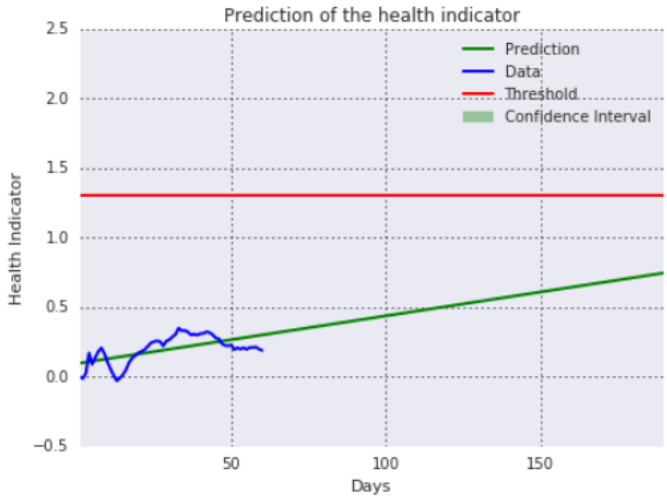


Figure B.19: Left accelerometer (X) condition prediction with regression model.

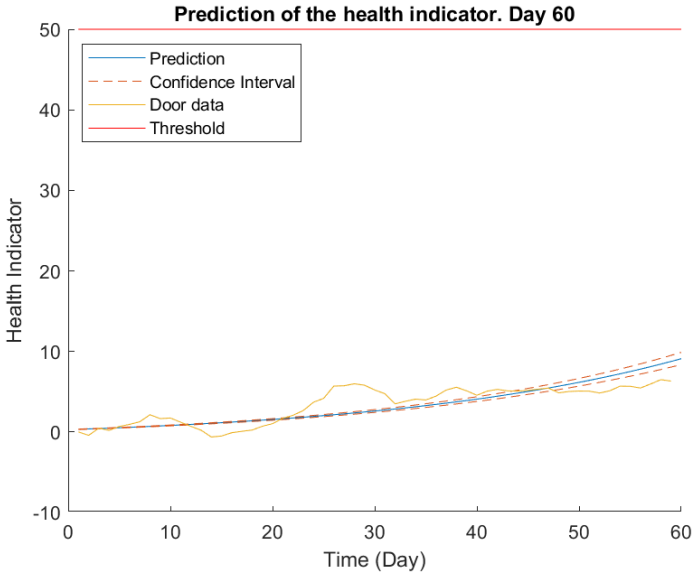


Figure B.20: Left accelerometer (X) condition prediction with degradation model.

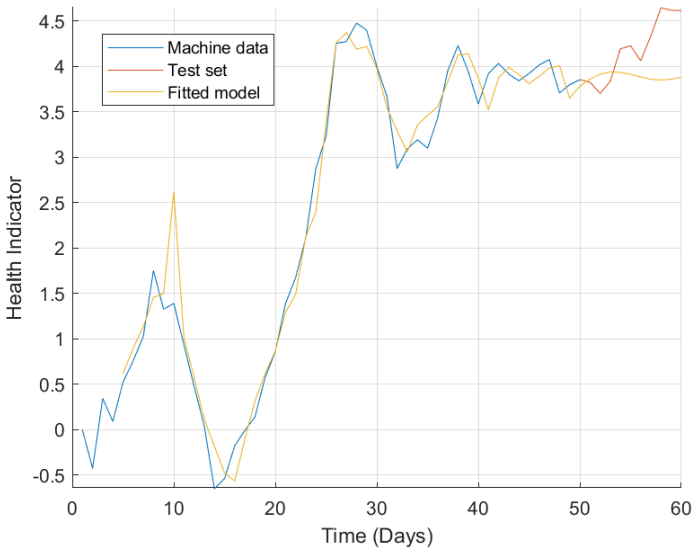


Figure B.21: Left accelerometer (X) condition prediction with neural network.

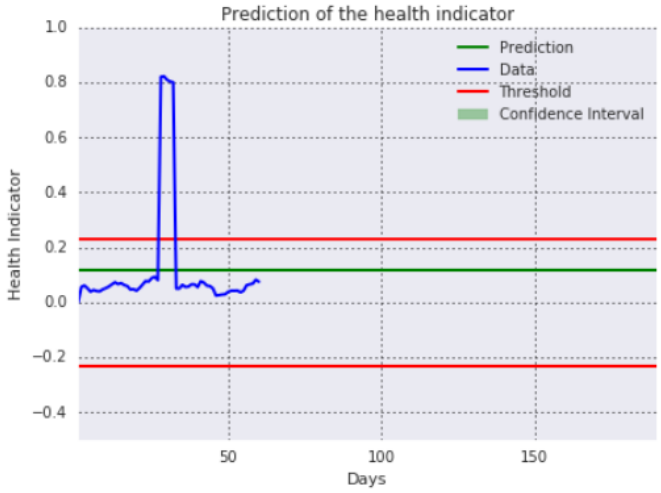


Figure B.22: Left accelerometer (Y) condition prediction with regression model.

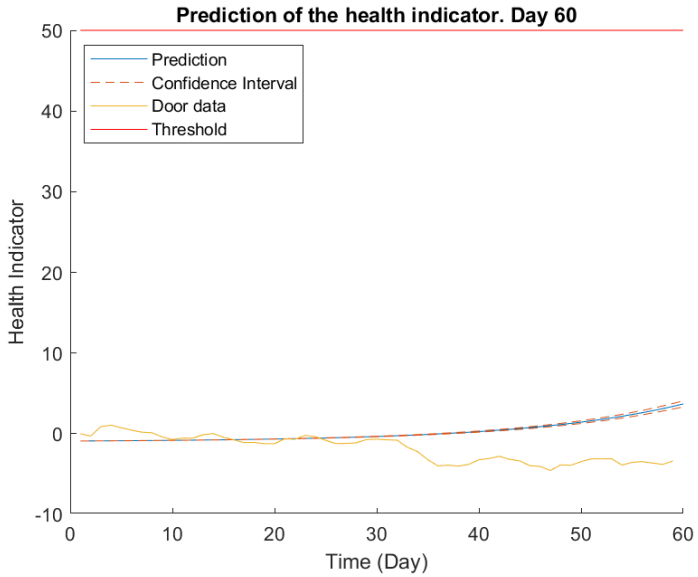


Figure B.23: Left accelerometer (Y) condition prediction with degradation model.

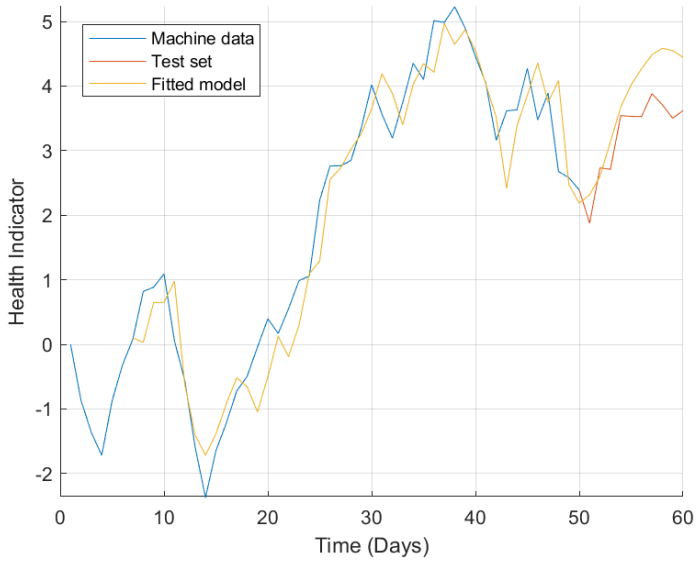


Figure B.24: Left accelerometer (Y) condition prediction with neural network.

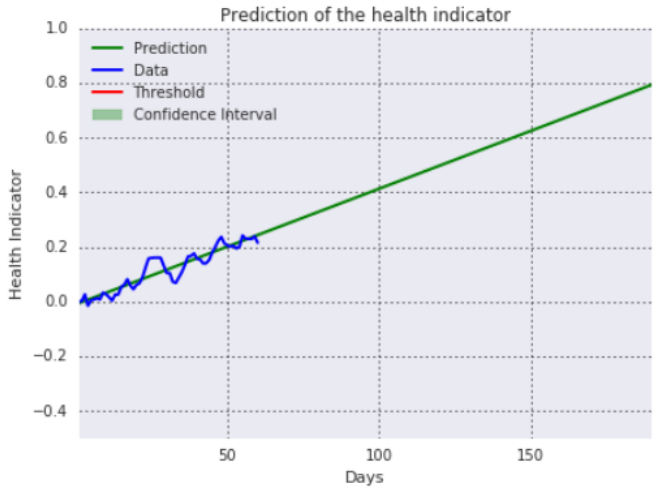


Figure B.25: Left accelerometer (Z) condition prediction with regression model.

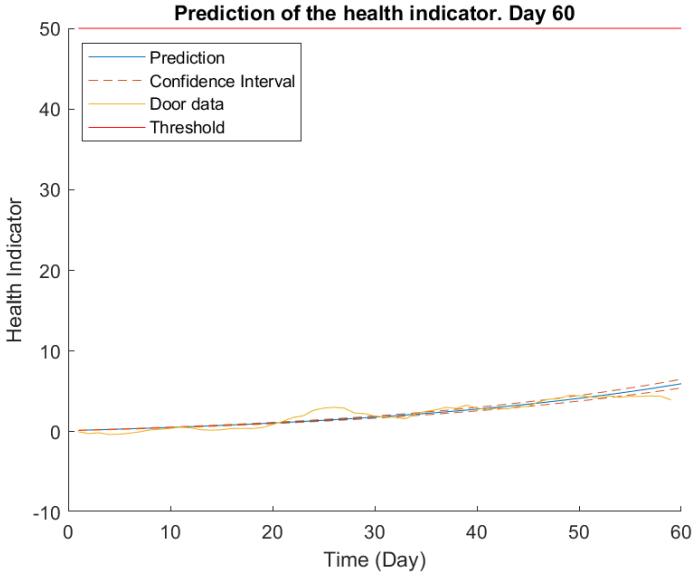


Figure B.26: Left accelerometer (Z) condition prediction with degradation model.

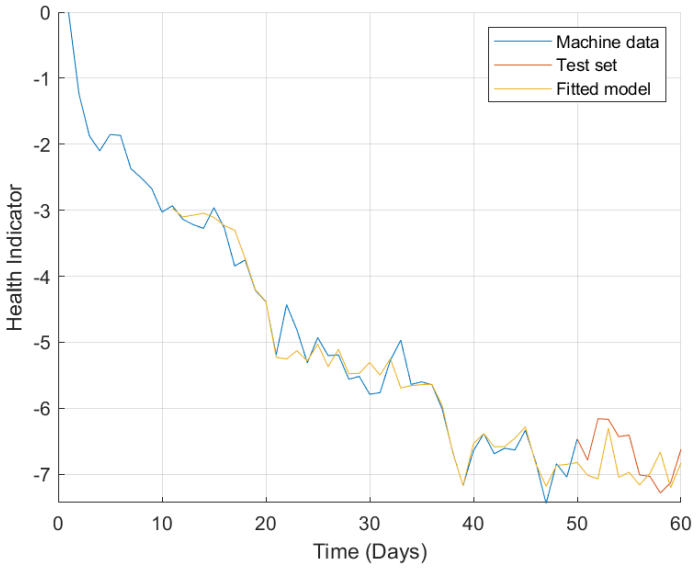


Figure B.27: Left accelerometer (Z) condition prediction with neural network.

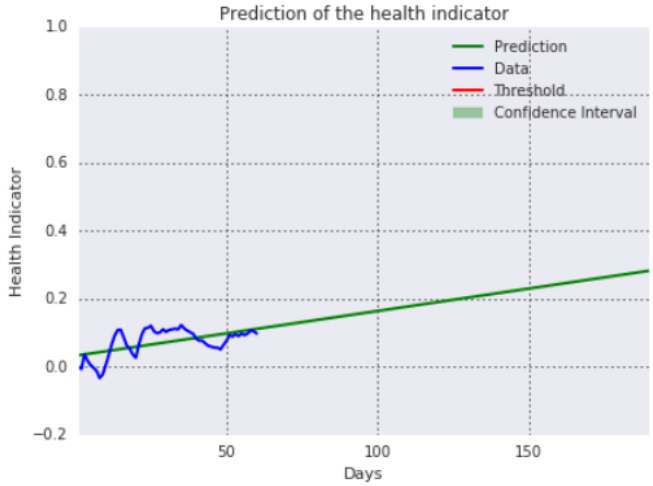


Figure B.28: Electric motor condition prediction with regression model.

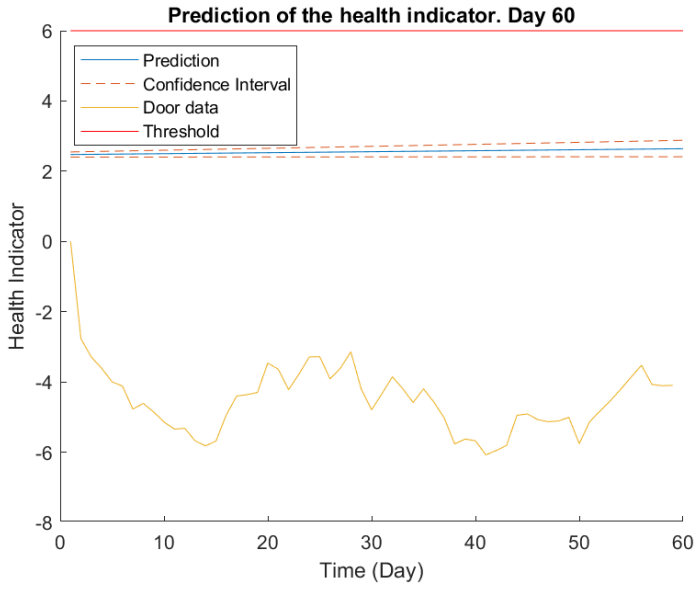


Figure B.29: Electric motor condition prediction with degradation model.

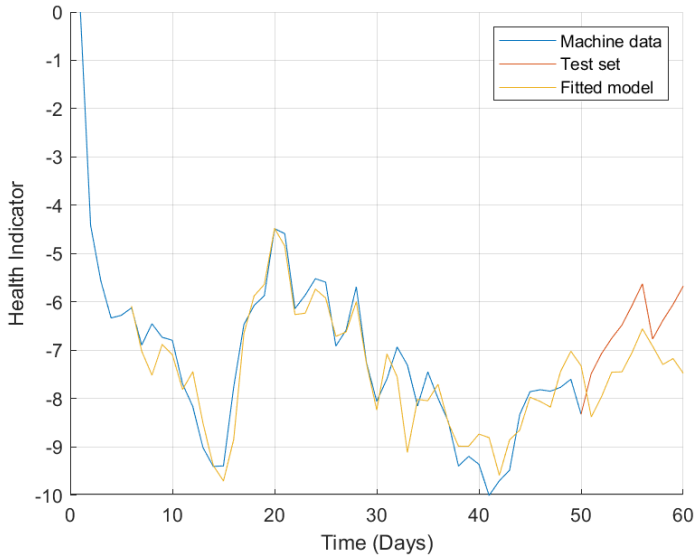


Figure B.30: Electric motor condition prediction with neural network.

C. Regression script

```
%matplotlib notebook
import MathEngine
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats
from scipy.stats import kurtosis
import pandas as pd
from sklearn.decomposition import PCA
from sklearn import preprocessing
from scipy.optimize import curve_fit
from scipy.signal import argrelextrema
from mpl_toolkits import mplot3d
import seaborn as sn
import time
import functools

deviceSerial = ###sensor serial number
inSensor     = ###sensor node numer
inChannel    = ###channel number [right frame, right accelerometer x, right accelerometer
               ###onderlever, left frame, left accelerometer x,left accelerometer y,left

data = list()
doorData = list()
maxDay = 60
measurementDay = 60 #4/12/2020

sampleRate = 256
sampleTime = 36
controlSensor     = ###door opening/closing signals
controlChannel    = ###channel number

def moving_avg(x, n):
    cumsum = np.cumsum(np.insert(x, 0, 0))
    return (cumsum[n:] - cumsum[:-n]) / float(n)
def get_column_array(df, column):
    expected_length = len(df)
    current_array = df[column].dropna().values
    if len(current_array) < expected_length:
        current_array = np.append(current_array, [''] * (expected_length - len(current_a
    return current_array
def stft(x, fs, frame_size, hop):
  """
  Perform STFT (Short-Time Fourier Transform).

  x: Input data.
  fs: Sampling rate.
  frame_size: Frame size.
  hop: Hop size
  """

  frame_samp = int(frame_size*fs)
  hop_samp = int(hop*fs)
  w = np.hanning(frame_samp) # Hanning window
  X = np.array([np.fft.fft(w*x[i:i+frame_samp])
                for i in range(0, len(x)-frame_samp, hop_samp)])
  return X
def timeavg(x):
    y = np.empty(len(x))
    for i in range(len(x)):
        y[i] = np.mean(x[i])
    return y
```

1

```
def rms(y):
    ms = 0
    for i in range(len(y)):
        ms = ms + y[i]**2
    ms = ms / len(y)
    rms = (ms)**0.5
    return rms
def monotonicity(x,m):
    monoList = 0
    for i in range(0,len(x)-1):
        monoList = monoList + (np.sign(x[i+1]-x[i])/m)
    return monoList

for day in range(maxDay):
    if (day != 4) and (day != 5) and (day != 6):
        start=1601999999950198000+day*86400000000000
        end=start+80000000000

        repo = TimeSeriesRepo(deviceSerial)

        inSeries = repo.getAllTimeSeries( inSensor, inChannel, startTime=start, endTime=
        data.append(inSeries.getData())
        print len(data[day]), 'Points'

        controlSeries = repo.getAllTimeSeries( controlSensor, controlChannel, startTime=
        doorData.append(controlSeries.getData())
    if day == 4:
        start=1602302399789547000
        end=start+80000000000

        repo = TimeSeriesRepo(deviceSerial)

        inSeries = repo.getAllTimeSeries( inSensor, inChannel, startTime=start, endTime=
        data.append(inSeries.getData())

        controlSeries = repo.getAllTimeSeries( controlSensor, controlChannel, startTime=
        doorData.append(controlSeries.getData())

        print len(data[day]), 'Points'
    if day == 5:
        a = np.zeros(len(data[day-1]))
        data.append(list(a))

        controlSeries = repo.getAllTimeSeries(controlSensor, controlChannel, startTime=s
        doorData.append(controlSeries.getData())

        print len(data[day]), 'Points'
    if day == 6:
        start=1602489599998559000
        end=start+80000000000

        repo = TimeSeriesRepo(deviceSerial)

        inSeries = repo.getAllTimeSeries( inSensor, inChannel, startTime=start, endTime=
        data.append(inSeries.getData())

        controlSeries = repo.getAllTimeSeries( controlSensor, controlChannel, startTime=
        doorData.append(controlSeries.getData())

        print len(data[day]), 'Points'
sensorData = pd.DataFrame(data)
```

```
sensorData.iloc[5] = np.nan
doorData = pd.DataFrame(doorData)

testData = sensorData
newSensorData = []
for testDay in range(maxDay):
    if doorData.iloc[testDay][0] > 0.8:
        startIndex = next((i for i, x in enumerate(doorData.iloc[testDay]) if x <0.2))-3
        if startIndex < 300:
            if doorData.iloc[testDay][startIndex+300] > 0.8:
                startIndex = next((i for i, x in enumerate(doorData.iloc[testDay][300:])
            elif doorData.iloc[testDay][300] < 0.2:
                nextIndex = next((i for i, x in enumerate(doorData.iloc[testDay][300:])
                startIndex = next((i for i, x in enumerate(doorData.iloc[testDay][nextIn
        endIndex = startIndex + sampleRate*sampleTime
        newSensorData.append(testData.iloc[testDay][startIndex:endIndex])
    elif doorData.iloc[testDay][0] < 0.2:
        nextIndex = next((i for i, x in enumerate(doorData.iloc[testDay]) if x >0.8))
        startIndex = next((i for i, x in enumerate(doorData.iloc[testDay][nextIndex:]) i
        endIndex = startIndex + sampleRate*sampleTime
        newSensorData.append(testData.iloc[testDay][startIndex:endIndex])
newSensorData = pd.DataFrame(newSensorData)
a = newSensorData.transpose()
b = pd.DataFrame({column: get_column_array(a, column) for column in a.columns})
newSensorData = b.iloc[0:sampleRate*sampleTime]
newSensorData = newSensorData.transpose()

for i in range(len(newSensorData)):
    for j in range(len(newSensorData.iloc[0])):
        if i != 5:
            newSensorData.iloc[i][j] = float(newSensorData.iloc[i][j])
        else:
            newSensorData.iloc[i][j] = np.nan
sensorData = newSensorData
sensorData.iloc[5] = np.nan

fft_s, freq_s = MathEngine.FFT(sensorData.iloc[0], inSeries.getSampleRate(), xMin=0, xMa
fft_e, freq_e = MathEngine.FFT(sensorData.iloc[-1], inSeries.getSampleRate(), xMin=0, xM
fft_s = fft_s/len(sensorData.iloc[0])/2
fft_e = fft_e/len(sensorData.iloc[-1])/2
plt.figure()
plt.plot(freq_s, fft_s/len(fft_s)/2,label='first day')
plt.plot(freq_e, fft_e/len(fft_e)/2,label='last day')
plt.xscale('log')
plt.legend(loc = 'upper right')

frame_size = 0.05  # with a frame size of 50 milliseconds
hop = 0.17

peakFreq = np.empty(len(sensorData))
SK = list()
fig = plt.figure()
ax = plt.axes(projection='3d')
for day in range(0,len(sensorData)):
    a = stft(sensorData.iloc[day],256,frame_size,hop)
    b = np.empty(len(a))
    if math.isnan(sensorData.iloc[day][0]):
        peakFreq[day] = np.nan
        SK.append(np.empty(len(SK[0])))
    else:
```

```
        fft, freq = MathEngine.FFT(sensorData.iloc[day], inSeries.getSampleRate(), xMin=
        peakFreq[day] = freq[np.where(fft == fft.max())]
        SK.append(timeavg((abs(a)**4))/(timeavg(abs(a)**2)**2)-2)
        skPlot = SK[day][0:len(freq)]
        ax.plot3D((day+1)*np.ones(len(freq)),freq,skPlot)

fig = plt.figure(figsize=(14,10))
ax = plt.axes(projection='3d')
for day in range(0,len(sensorData)):
    fft, freq = MathEngine.FFT(sensorData.iloc[day], inSeries.getSampleRate(), xMin=0, x
    fft = fft/len(sensorData.iloc[day])/2
    ax.plot3D(day*np.ones(len(freq)), freq, fft)

SK = pd.DataFrame(SK)
SK.iloc[5] = np.nan
sensorData.reset_index().values
features = list()
for i in range(len(sensorData)):
    features.append(list())
for i in range(len(sensorData)):
    features[i].append(stats.tmean(sensorData.iloc[i]))
    features[i].append(np.std(sensorData.iloc[i]))
    features[i].append(stats.skew(sensorData.iloc[i]))
    features[i].append(stats.kurtosis(sensorData.iloc[i]))
    features[i].append(np.ptp(sensorData.iloc[i]))
    features[i].append(rms(sensorData.iloc[i]))
    features[i].append(max(sensorData.iloc[i])/rms(sensorData.iloc[i]))
    features[i].append(rms(sensorData.iloc[i])/abs(stats.tmean(sensorData.iloc[i])))
    features[i].append(max(sensorData.iloc[i])/stats.tmean(sensorData.iloc[i]))
    features[i].append(max(sensorData.iloc[i])/(stats.tmean(sensorData.iloc[i]))**2)

    features[i].append(np.mean(SK.iloc[i]))
    features[i].append(np.std(SK.iloc[i]))
    features[i].append(stats.skew(SK.iloc[i]))
    features[i].append(stats.kurtosis(SK.iloc[i]))
    features[i].append(peakFreq[i])

features = pd.DataFrame(features)
for i in range(len(features)):
    for j in range(len(features.iloc[i])):
        if math.isnan(features.iloc[i][j]):
            features.iloc[i][j] = (features.iloc[i+1][j]+features.iloc[i-1][j])/2

featureTableSmooth = []
windowSize = 5
for i in range(0,len(features.iloc[i])):
    featureTableSmooth.append(moving_avg(np.array(features[i]),windowSize))

newFeaturetable = pd.DataFrame(featureTableSmooth[0])
for i in range(1,len(featureTableSmooth)):
    newFeaturetable[i]= (pd.DataFrame(featureTableSmooth[i]))
featureTableSmooth = newFeaturetable
newLength = len(newFeaturetable)

for i in range(0,windowSize-1):
        featureTableSmooth = featureTableSmooth.append(pd.DataFrame(np.ones(len(featureT
        ,ignore_index=True)
for i in range(len(featureTableSmooth)-1,-1,-1):
        featureTableSmooth.iloc[i] = featureTableSmooth.iloc[i-(windowSize-1)]
for i in range(0,windowSize-1):
        featureTableSmooth.iloc[i] = features.iloc[i]
```

```
features = pd.DataFrame(featureTableSmooth)
monoTable = np.empty(len(features.iloc[0]))
m = len(data)-1
for feature in range(len(features.iloc[0])):
        monoTable[feature] = monotonicity(features[feature],m)
for i in range(len(monoTable)):
    monoTable[i] = abs(monoTable[i])
monoTable= pd.DataFrame(monoTable)

indices = list()
k = 0
if np.mean(monoTable[0]) > 0.3:
        limit = 0.3
else:
    limit = np.mean(monoTable[0])
for i in range(len(monoTable[0])):
    if monoTable[0][i] > limit:
        indices.append(i)
        k = k+1

trainData = pd.DataFrame(features)[indices]
dataLabels = np.array(['Mean', 'Std', 'Skewness', 'Kurtosis',
       'Peak_to_peak','RMS', 'Crest_Factor','Shape_Factor','Impulse_Factor','Margin_Fact
        "SKKurtosis",'Natural_Frequency'])
trainData.columns = dataLabels[indices]
normalized = preprocessing.normalize(trainData.transpose())

trainNormal = pd.DataFrame(normalized.transpose())
trainNormal.columns = dataLabels[indices]

pca_data = PCA(n_components=2)
principalComponents_data = pca_data.fit_transform(trainNormal)
principal_data_Df = pd.DataFrame(data = principalComponents_data , columns = [0, 1])
print('Explained variation per principal component: {}'.format(pca_data.explained_varian

x_data = principal_data_Df[0]
y_data = principal_data_Df[1]
t = range(len(x_data))
t = [x + 1 for x in t]
plt.scatter(x_data, y_data, c=t,s=100)
plt.grid(color='k', linestyle='dotted', linewidth=1)
cbar = plt.colorbar()
plt.ylabel("second principal component")
plt.xlabel("first principal component")
cbar.set_label("Days")
plt.show()

pcas = list()
pcas.append(abs(pca_data.explained_variance_ratio_))
index = pcas.index(max(pcas))
healthIndicator = np.empty(len(principalComponents_data))
for i in range(len(principalComponents_data)):
    healthIndicator[i] = principalComponents_data[i][index]

healthIndicator = healthIndicator-healthIndicator[0]
threshold1 = 1.52##experimental value
threshold2 = -threshold1
ydata = np.empty(len(healthIndicator))
plt.plot(healthIndicator,linewidth=2.0)
plt.grid(color='k', linestyle='dotted', linewidth=1)
```

```
if healthIndicator[-1] < 0:
        healthIndicator = -healthIndicator

def func(x, a, b, c):
    return abs(a)*np.exp(abs(b)*x)+c
xdata = range(len(healthIndicator))
xdata = [x + 1 for x in xdata]
for day in range(0,len(healthIndicator)):
    ydata[day] = healthIndicator[day]
popt, pcov = curve_fit(func, xdata, ydata,p0=[0,0,0],maxfev = 8000)
sigma_ab = np.sqrt(np.diagonal(pcov))

x = np.linspace(0,len(healthIndicator)+200,1000)
x = [x + 1 for x in x]
x = np.array(x)
y = func(x, *popt)

plt.figure(figsize=(7,5))

plt.plot(x,y,'g',label='Prediction',linewidth=2.0)
plt.plot(xdata, ydata, 'b', label='Data',linewidth=2.0)
plt.hlines(threshold1,xmin=x[0],xmax=x[-1],color='r',label='Threshold',linewidth=2.0)
plt.hlines(threshold2,xmin=x[0],xmax=x[-1],color='r',linewidth=2.0)
plt.grid(color='k', linestyle='dotted', linewidth=1)
plt.ylim(-0.5, 1)
plt.xlim(1, maxDay+130)


bound_upper = func(x, *(popt + sigma_ab))
bound_lower = func(x, *(popt - sigma_ab))
plt.fill_between(x, bound_lower, bound_upper,
                 color = 'green', alpha = 0.35,label='Confidence Interval')

leg = plt.legend(loc="best")


plt.title('Prediction of the health indicator')
plt.xlabel('Days')
plt.ylabel('Health Indicator')



plt.show(sn)

RUL1 = np.log((threshold1-popt[2])/popt[0])/popt[1] - len(healthIndicator)
RUL2 = np.log((threshold2-popt[2])/popt[0])/popt[1] - len(healthIndicator)
RUL = max(RUL1,RUL2)
print RUL
```

D. Predictive maintenance toolbox

```
%% START
timeUnit = 'day';
totalDay = 60;
for i = 1:totalDay
    tday(i) =  datetime(2020,10,i+5,18,0,0);
end

varIndex = 1;% 1 - 10 to select one of the variables to make prediction;
% [left frame, left underlever, right frame, left accelerometer x, left
% accelerometer y, left accelerometer z, right accelerometer x, right
% accelerometer y, right accelerometer z, current ]
%% IMPORT DATA
data = importdata('Path to folder');
%% VARIABLE SELECTION
data = table(data{:,varIndex});
%% VIBRATION ANALYSIS
fs = 256;

tstart = 0;
figure
hold on
dataNum = 0;
for k = 1:totalDay
    v = data.Var1(1+dataNum:dataLength+dataNum);
    t = tstart + (1:length(v))/fs;
    if ~isnan(v(1))
        plot(t(1:length(v)), v(1:length(v)))
    end
    tstart = t(dataLength);
    dataNum = (k)*dataLength;
end
hold off
xlabel('Time (s), 36 seconds per day, 60 days in total')
ylabel(sprintf('Acceleration (g)'))

clear ind t v tstart k
%% SPECTRAL KURTOSIS
colors = parula(totalDay);
DATA = table;
figure
hold on
day = 1;

dataNum = 0;
for day = 1:totalDay
    data2add = table;

    v = data.Var1(1+dataNum:dataLength+dataNum);
    if  ~isnan(v)
        [SK, F] = pkurtosis(v, fs);
        data2add.SpectralKurtosis = {table(F, SK)};
        plot3(F, day*ones(size(F)), SK, 'Color', colors(day, :))
        DATA(day,1) = data2add.SpectralKurtosis;
```

1

```
    end
    dataNum = (day)*dataLength;
end
hold off
xlabel('Frequency (Hz)')
ylabel('Time (day)')
zlabel('Spectral Kurtosis')
grid on
view(-45, 30)
cbar = colorbar;
ylabel(cbar, 'Fault Severity (0 - healthy, 1 - faulty)')
%% MISSING VALUES
SK6 = (DATA.Var1{7,1}.SK+DATA.Var1{5,1}.SK)/2
F6 = (DATA.Var1{7,1}.F+DATA.Var1{5,1}.F)/2
DAT6 = table(F, SK)
DATA.Var1{6,1}=DAT6
clear SK6 F6 DAT6
%% FEATURE EXTRACTION
dataNum = 0;
features = table;
day = 1;

dataNum = 0;
for day = 1:totalDay
    v = data.Var1(1+dataNum:dataLength+dataNum);
    SK = DATA.Var1{day,1}.SK;
    % Time Domain Features
    features.mean(day) = mean(v);
    features.std(day) = std(v);
    features.skewness(day) = skewness(v);
    features.kurtosis(day) = kurtosis(v);
    features.peak2peak(day) = peak2peak(v);
    features.rms(day) = rms(v);
    features.crestfactor(day) = max(v)/features.RMS(day);
    features.shapefactor(day) = features.RMS(day)/mean(abs(v));
    features.impulsefactor(day) = max(v)/mean(abs(v));
    features.marginfactor(day) = max(v)/mean(abs(v))^2;
    features.energy(day) = sum(v.^2);

    % Spectral features
    features.smean(day) = mean(SK);
    features.sstd(day) = std(SK);
    features.sskewness(day) = skewness(SK);
    features.skurtosis(day) = kurtosis(SK);
    dataNum = (day)*dataLength;
end
clear dataNum data2add SK F data
%% MISSING VALUES
    features.mean(6) = (features.mean(7)+features.mean(5))/2;
    features.std(6) = (features.std(7)+features.std(5))/2;
    features.skewness(6) = (features.skewness(7)+features.skewness(5))/2;
    features.kurtosis(6) = (features.kurtosis(7)+features.kurtosis(5))/2;
    features.peak2peak(6) = (features.peak2peak(7)+features.peak2peak(5))/2;
    features.rms(6) = (features.rms(7)+features.rms(5))/2;
```

```
    features.crestfactor(6) = (features.srestfactor(7)+features.crestfactor(5))/2;
    features.shapefactor(6) = (features.shapefactor(7)+features.shapefactor(5))/2;
    features.impulsefactor(6) = (features.impulsefactor(7)+features.impulsefactor(5))/2;
    features.marginfactor(6) = (features.marginfactor(7)+features.marginfactor(5))/2;
    features.energy(6) = (features.energy(7)+features.energy(5))/2;

    % Spectral features
    features.smean(6) = (features.smean(7)+features.smean(5))/2;
    features.sstd(6) = (features.sstd(7)+features.sstd(5))/2;
    features.sskewness(6) = (features.sskewness(7)+features.sskewness(5))/2;
    features.skurtosis(6) = (features.skurtosis(7)+features.skurtosis(5))/2;
%% FEATURE TABLE
tableoffeatures = gather(tall(features));
for i = 1:totalDay
    tableoffeatures.Date(i)=tday(i)
end
tableoffeatures = table2timetable(tableoffeatures)
clear numData tday varIndex v fs dataLength
%% FEATURE POSTPROCESSING
smoothFeatureTable = table;
variableNames = tableoffeatures.Properties.VariableNames;
smoothFeatureTable = varfun(@(x) movmean(x, [5 0]), tableoffeatures);
smoothFeatureTable.Properties.VariableNames = variableNames;

smoothFeatureTable.Date = tableoffeatures.Date;

%% TRAINING DATA
trainSet = featureTableSmooth(1:50, :);
%% FEATURE RANKING
featureImportance = monotonicity(trainSet, 'WindowSize', 0);
helperSortedBarPlot(featureImportance, 'Monotonicity');
%% FEATURE SELECTION
selectedTrainData = trainSet(:, featureImportance{:,:}>0.2);
selectedFeatures = smoothFeatureTable(:, featureImportance{:,:}>0.2)
%% DIMENSION REDUCTION
trainAvg = mean(selectedTrainData{:,:});
trainStd = std(selectedTrainData{:,:});
rainData = (selectedTrainData{:,:} - trainAvg)./trainStd;
coef = pca(trainDataNormalized);

PCA_1 = (selectedFeatures{:,:} - trainAvg) ./ trainStd * coef(:, 1);
PCA_2 = (selectedFeatures{:,:} - trainAvg) ./ trainStd * coef(:, 2);

figure
numData = size(featureTable, 1);
scatter(PCA_1, PCA_2, [], 1:numData, 'filled')
xlabel('PCA 1')
ylabel('PCA 2')
cbar = colorbar;
ylabel(cbar, ['Time (' timeUnit ')'])
%% HEALTH INDICATOR
healthIndicator = PCA_1;

healthIndicator = healthIndicator - healthIndicator(1);
```

```
threshold = 6;%%experimental data

figure
hold on
plot(selectedFeatures.Date, healthIndicator, '-o')
xlabel('Time')
title('Health Indicator')
%% PREDICTION
mdl = exponentialDegradationModel(...
    'Theta', 1, ...
    'ThetaVariance', 1e6, ...
    'Beta', 1, ...
    'BetaVariance', 1e6, ...
    'Phi', -1, ...
    'NoiseVariance', (0.1*threshold/(threshold + 1))^2, ...
    'SlopeDetectionLevel', 0.05);
%% PLOTTING
% Keep records at each iteration
totalDay = length(healthIndicator) - 1;
estRULs = zeros(totalDay, 1);
trueRULs = zeros(totalDay, 1);
CIRULs = zeros(totalDay, 2);
pdfRULs = cell(totalDay, 1);

% Create figures and axes for plot updating
figure
ax1 = subplot(2, 1, 1);
ax2 = subplot(2, 1, 2);

for currentDay = 1:totalDay

    % Update model parameter posterior distribution
    update(mdl, [currentDay healthIndicator(currentDay)])

    % Predict Remaining Useful Life
    [estRUL, CIRUL, pdfRUL] = predictRUL(mdl, ...
                                         [currentDay healthIndicator(currentDay)], ...
                                         threshold);
    trueRUL = totalDay - currentDay + 1;

    % Updating RUL distribution plot
    helperPlotTrend(ax1, currentDay, healthIndicator, mdl, threshold, timeUnit);
    helperPlotRUL(ax2, trueRUL, estRUL, CIRUL, pdfRUL, timeUnit)

    % Keep prediction results
    estRULs(currentDay) = estRUL;
    trueRULs(currentDay) = trueRUL;
    CIRULs(currentDay, :) = CIRUL;
    pdfRULs{currentDay} = pdfRUL;

    % Pause 0.1 seconds to make the animation visible
    pause(0.3)
end
%% PERFORMANCE
```

E. Neural Network

```
function [monoList] = monotonicity(x)
    [m, n] = size(x);
    monoList = zeros(1,n);
    for k = 1:n
        for j = 1:m-1
            monoList(k) = monoList(k) + sign(x(j+1,1)-x(j,1))/m;
        end
    end

timeUnit = 'day';
totalDay = 60;
for i = 1:totalDay
    tday(i) =  datetime(2020,10,i+5,18,0,0);
end

varIndex = 1;% 1 - 10 to select one of the variables to make prediction;
% [left frame, left underlever, right frame, left accelerometer x, left
% accelerometer y, left accelerometer z, right accelerometer x, right
% accelerometer y, right accelerometer z, current ]
%% IMPORT DATA
data = importdata('Path to folder');
%% VARIABLE SELECTION
data = table(data{:,varIndex});
%% VIBRATION ANALYSIS
fs = 256;

tstart = 0;
figure
hold on
dataNum = 0;
for k = 1:totalDay
    v = data.Var1(1+dataNum:dataLength+dataNum);
    t = tstart + (1:length(v))/fs;
    if ~isnan(v(1))
        plot(t(1:length(v)), v(1:length(v)))
    end
    tstart = t(dataLength);
    dataNum = (k)*dataLength;
end
hold off
xlabel('Time (s), 36 seconds per day, 60 days in total')
ylabel(sprintf('Acceleration (g)'))

clear ind t v tstart k
%% SPECTRAL KURTOSIS
colors = parula(totalDay);
DATA = table;
figure
hold on
day = 1;

dataNum = 0;
for day = 1:totalDay
```

1

```
    data2add = table;

    v = data.Var1(1+dataNum:dataLength+dataNum);
    if  ~isnan(v)
        [SK, F] = pkurtosis(v, fs);
        data2add.SpectralKurtosis = {table(F, SK)};
        plot3(F, day*ones(size(F)), SK, 'Color', colors(day, :))
        DATA(day,1) = data2add.SpectralKurtosis;
    end
    dataNum = (day)*dataLength;
end
hold off
xlabel('Frequency (Hz)')
ylabel('Time (day)')
zlabel('Spectral Kurtosis')
grid on
view(-45, 30)
cbar = colorbar;
ylabel(cbar, 'Fault Severity (0 - healthy, 1 - faulty)')
%% MISSING VALUES
SK6 = (DATA.Var1{7,1}.SK+DATA.Var1{5,1}.SK)/2
F6 = (DATA.Var1{7,1}.F+DATA.Var1{5,1}.F)/2
DAT6 = table(F, SK)
DATA.Var1{6,1}=DAT6
clear SK6 F6 DAT6
%% FEATURE EXTRACTION
dataNum = 0;
features = table;
day = 1;

dataNum = 0;
for day = 1:totalDay
    v = data.Var1(1+dataNum:dataLength+dataNum);
    SK = DATA.Var1{day,1}.SK;
    % Time Domain Features
    features.mean(day) = mean(v);
    features.std(day) = std(v);
    features.skewness(day) = skewness(v);
    features.kurtosis(day) = kurtosis(v);
    features.peak2peak(day) = peak2peak(v);
    features.rms(day) = rms(v);
    features.crestfactor(day) = max(v)/features.RMS(day);
    features.shapefactor(day) = features.RMS(day)/mean(abs(v));
    features.impulsefactor(day) = max(v)/mean(abs(v));
    features.marginfactor(day) = max(v)/mean(abs(v))^2;
    features.energy(day) = sum(v.^2);

    % Spectral features
    features.smean(day) = mean(SK);
    features.sstd(day) = std(SK);
    features.sskewness(day) = skewness(SK);
    features.skurtosis(day) = kurtosis(SK);
    dataNum = (day)*dataLength;
end
```

```
clear dataNum data2add SK F data
%% MISSING VALUES
    features.mean(6) = (features.mean(7)+features.mean(5))/2;
    features.std(6) = (features.std(7)+features.std(5))/2;
    features.skewness(6) = (features.skewness(7)+features.skewness(5))/2;
    features.kurtosis(6) = (features.kurtosis(7)+features.kurtosis(5))/2;
    features.peak2peak(6) = (features.peak2peak(7)+features.peak2peak(5))/2;
    features.rms(6) = (features.rms(7)+features.rms(5))/2;
    features.crestfactor(6) = (features.srestfactor(7)+features.crestfactor(5))/2;
    features.shapefactor(6) = (features.shapefactor(7)+features.shapefactor(5))/2;
    features.impulsefactor(6) = (features.impulsefactor(7)+features.impulsefactor(5))/2;
    features.marginfactor(6) = (features.marginfactor(7)+features.marginfactor(5))/2;
    features.energy(6) = (features.energy(7)+features.energy(5))/2;

    % Spectral features
    features.smean(6) = (features.smean(7)+features.smean(5))/2;
    features.sstd(6) = (features.sstd(7)+features.sstd(5))/2;
    features.sskewness(6) = (features.sskewness(7)+features.sskewness(5))/2;
    features.skurtosis(6) = (features.skurtosis(7)+features.skurtosis(5))/2;
%% FEATURE TABLE
tableoffeatures = gather(tall(features));
for i = 1:totalDay
    tableoffeatures.Date(i)=tday(i)
end
tableoffeatures = table2timetable(tableoffeatures)
clear numData tday varIndex v fs dataLength
%% FEATURE POSTPROCESSING
smoothFeatureTable = table;
variableNames = tableoffeatures.Properties.VariableNames;
smoothFeatureTable = varfun(@(x) movmean(x, [5 0]), tableoffeatures);
smoothFeatureTable.Properties.VariableNames = variableNames;

smoothFeatureTable.Date = tableoffeatures.Date;

%% FEATURE IMPORTANCE
m = totalDay
featureImportance(1,1) = monotonicity(smoothFeatureTable.mean)
featureImportance(1,2) = monotonicity(smoothFeatureTable.std)
featureImportance(1,3) = monotonicity(smoothFeatureTable.skewness)
featureImportance(1,4) = monotonicity(smoothFeatureTable.kurtosis)
featureImportance(1,5) = monotonicity(smoothFeatureTable.peak2peak)
featureImportance(1,6) = monotonicity(smoothFeatureTable.rms)
featureImportance(1,7) = monotonicity(smoothFeatureTable.crestfactor)
featureImportance(1,8) = monotonicity(smoothFeatureTable.shapefactor)
featureImportance(1,9) = monotonicity(smoothFeatureTable.impulsefactor)
featureImportance(1,10) = monotonicity(smoothFeatureTable.marginfactor)
featureImportance(1,11) = monotonicity(smoothFeatureTable.energy)
featureImportance(1,12) = monotonicity(smoothFeatureTable.smean)
featureImportance(1,13) = monotonicity(smoothFeatureTable.sstd)
featureImportance(1,14) = monotonicity(smoothFeatureTable.sskewness)
featureImportance(1,15) = monotonicity(smoothFeatureTable.skurtosis)

%% TRAINING DATA
trainset = smoothFeatureTable(1:end, :);
```

```
%% FEATURE SELECTION

featureImportance = abs(featureImportance')

idx = find(featureImportance > 0.3)
trainSelection = trainset(:, featureImportance(:,:)>0.3);
selectedFeatures = smoothFeatureTable(:, featureImportance(:,:)>0.3)

if sum(length(idx)) < 2
    idx = find(featureImportance > mean(featureImportance))
    trainSelection = trainset(:, featureImportance(:,:)>mean(featureImportance));
    selectedFeatures = smoothFeatureTable(:, featureImportance(:,:)>mean(featureImportance))
end
trainSelection = table2array(trainSelection);
%% DIMENSION REDUCTION
trainAvg = mean(trainSelection(:,:));
trainStd = std(trainSelection(:,:));
normalTrainData = (trainSelection(:,:) - trainAvg)./trainStd;
coefs = pca(normalTrainData);

PCA_1 = (selectedFeatures{:,:} - trainAvg) ./ trainStd * coefs(:, 1);
PCA_2 = (selectedFeatures{:,:} - trainAvg) ./ trainStd * coefs(:, 2);

figure
numData = size(tableoffeatures, 1);
scatter(PCA1, PCA_2, [], 1:numData, 'filled')
xlabel('PCA 1')
ylabel('PCA 2')
cbar = colorbar;
ylabel(cbar, ['Time (' timeUnit ')'])

pcas = [PCA1 PCA_2];
idx = find(max(abs(monotonicity2(pcas))));
healthIndicator = pcas(:,idx);

figure
plot(1:length(healthIndicator), healthIndicator, '-o')
xlabel('Time')
title('Health Indicator')

healthIndicator = healthIndicator - healthIndicator(1);
threshold = healthIndicator(end-1);
clear variableNames trainDataSelected trainDataNormalized trainData pcas PCA1 PCA2 featureSelec
%% NEURAL NETWORK FITTER
days = 1:totalDay;
days = days';
%% NN TIME SERIES 1
delay = 5
T = num2cell(healthIndicator(1:50)');
net = narnet(1:delay,10);

[Xs,Xi,Ai,Ts] = preparets(net,{},{},T);
net = train(net,Xs,Ts,Xi,Ai);
view(net)
```

```
%% 2
[Y,Xf,Af] = net(Xs,Xi,Ai);
perf = perform(net,Ts,Y)
%% 3
[netc,Xic,Aic] = closeloop(net,Xf,Af);
view(netc)
%% 4
y2 = netc(cell(0,10),Xic,Aic)
%% 5
y2 = cell2mat(y2);
Y = cell2mat(Y)
%% TIME SERIES PLOTTING
pred = [Y,y2]
figure
hold on
plot(days(1:50),healthIndicator(1:50))
plot(days(50:end),healthIndicator(50:end))
daysPred = 1:length(pred);
plot(daysPred+delay,pred)
grid on
hold off
xlabel('Time (Days)')
ylabel('Health Indicator')
ylim([min(healthIndicator),max(healthIndicator)])
legend('Machine data','Test set','Fitted model');
fprintf('Remaining useful life = %d days. \n',min((pred)> threshold)-totalDay )
error = 100*((pred(50-delay:end) - healthIndicator(50:end)') ./ healthIndicator(50:end)');
%% CLEAR
clear colors i idx m timeUnit coef cbar breaktime breakpoint sdTrain meanTrain
```

Bibliography

[1] Predictive maintenance market analysis report by solution, by service, by deployment, by enterprise size, by end use, by region and segment forecasts from 2019 to 2025. Technical report, Grand View Research, November 2019.

[2] Bernard Boser, Isabelle Guyon, and Vladimir Vapnik. Pattern recognition system using support vectors, May 1996.

[3] Bryan Christiansen. 3 main types of maintenance strategies, 2020. https://limblecmms.com/blog/3-main-types-of-maintenance-strategies/#, accessed: 01.09.2020.

[4] Bryan Christiansen. A complete guide to predictive maintenance, 2020. https://limblecmms.com/blog/predictive-maintenance/#, accessed: 01.09.2020.

[5] Jamie Coble and J. Hines. Identifying optimal prognostic parameters from data: A genetic algorithms approach. *Annual Conference of the Prognostics and Health Management Society, San Diego, CA, September*, 01 2009.

[6] Ilesanmi Daniyan, Khumbulani Mpofu, Moses Oyesola, Boitumelo Ramatsetse, and Adefemi Adeodu. Artificial intelligence for predictive maintenance in the railcar learning factories. *Procedia Manufacturing*, 45:13–18, 2020. https://doi.org/10.1016/j.promfg.2020.04.032.

[7] Nagi Gebraeel. Sensory-updated residual life distributions for components with exponential degradation patterns. *Automation Science and Engineering, IEEE Transactions on*, 3:382 – 393, 11 2006.

[8] Mark Haarman, Michel Mulders, and Costas Vassiliadis. Predictive maintenance 4.0: Predict the unpredictable. Technical report, pwc and mainnovation, June, 2017.

[9] Micharl Decourcy Hinds. Preventive Maintenance: A Checklist. *The New York Times*, page 14, 1895.

[10] I.A.Daniyan, K. Mpofu, and A.O. Adeodu. Development of a diagnostic and prognostic tool for predictive maintenance in the railcar industry. *Procedia CIRP*, 90:109–114, 2020. https://doi.org/10.1016/j.procir.2020.02.001.

[11] Veronica Jaramillo Jimenez, Nouredinne Bouhmala, and Anne Haugen Gausdal. Developing a predictive maintenance model for vessel machinery. *Journal of Ocean Engineering and Science*, page 14, 2020. https://doi.org/10.1016/j.joes.2020.03.003.

[12] Patrick Killeen, Bo Ding, Iluju Kiringa, and Tet Yeap. IoT-based predictive maintenance for fleet management. *Procedia Computer Science*, 151:607–613, 2019. https://doi.org/10.1016/j.procs.2019.04.184.

[13] MathWorks. Overcoming four common obstacles to predictive maintenance with matlab and simulink. Technical report, MathWorks, 2018.

[14] MathWorks. Introduction to predictive maintenance with matlab. Technical report, MathWorks, 2019.

[15] MathWorks. Predictive maintenance: Estimating remaining useful life with matlab. Technical report, MathWorks, 2019.

[16] MathWorks. Predictive maintenance: Extracting condition indicators with matlab. Technical report, MathWorks, 2019.

[17] MATLAB. Wind turbine high-speed bearing prognosis, 2020. https://nl.mathworks.com/help/predmaint/ug/wind-turbine-high-speed-bearing-prognosis.html, accessed: 18.11.2020.

[18] R. Keith Mobley. *An Introduction to Predictive Maintenance*. Elsevier, 2002.

[19] Gordon Moore. Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, april 19, 1965, pp.114 ff. *Solid-State Circuits Newsletter, IEEE*, 11:33 – 35, 10 2006.

[20] Srikanth Namuduri, Barath Narayanan Narayanan, Venkata Salini Priyamvada Davuluru, Lamar Burton, and Shekhar Bhansali. Review—deep learning methods for sensor based predictive maintenance and future perspectives for electrochemical sensors. *Journal of The Electrochemical Society*, 167, 01 2020.

[21] F.S. Nowlan and H.F. Heap. Reliability-centered maintenance. *United airlines*, 1978.

[22] TÜV Rheinland. Emit optimisation – getting more out of existing equipment for less. *The Newsletter of Risktec Solutions*, pages 4–5, 2017.

[23] Cornelius Scheffer and Paresh Girdhar. *Practical Machinery Vibration Analysis and Predictive Maintenance*. Elsevier, 2004.

[24] SensorCloud. Good vibrations for machine health monitoring: Maximizing uptime with intelligent vibration monitoring and predictive analytics, 2020. https://sensorcloud.com/static/files/documents/SolutionBrief_SCVibration.pdf, accessed: 28.09.2020.

[25] SensorCloud. Machine monitoring applications in the oil and gas industry: Real-time health monitoring of high value assetss, 2020. https://sensorcloud.com/static/files/documents/SolutionBrief_SCOil&Gas.pdf, accessed: 28.09.2020.

[26] Sandro Sperandei. Understanding logistic regression analysis. *Biochemia medica*, 24:12–8, 02 2014.

[27] G. P. Sullivan, R. Pugh, A. P. Melendez, and W. D. Hunt. O&m best practices guide, release 3.0. Technical report, United States Department of Energy, August 2010.

[28] Ventura Systems. Ventura systems innovative bus door systems, 2020. https://www.venturasystems.com/about-us, accessed: 01.09.2020.

[29] Darren Whitaker, David Egan, Eoin OBrien, and David Kinnear. Application of multivariate data analysis to machine power measurements as a means of tool life predictive maintenance for reducing product waste, 02 2018.

[30] Haiyue Wu, Aihua Huang, and John W. Sutherland. Avoiding Environmental Consequences of Equipment Failure via an LSTM-Based Model for Predictive Maintenance. *Procedia Manufacturing*, 43:666–673, 2020. https://doi.org/10.1016/j.promfg.2020.02.131.