# Data Pipelines and Edge Analysis in Engine Installations

Samuel Kekäläinen

Master's thesis
Supervisor: Kim Westerlund, Edupower Oy Ab
Examinator: Prof. Margareta Björklund-Sänkiaho
Energy Technology, Vasa
Study programme in Chemical Engineering
Faculty of Science and Engineering
Åbo Akademi University
June 15, 2021

# ABSTRAKT

Intresset för datainsamling och analys har ökat i samma takt som processorkraft har ökat och nya maskininlärningsalgoritmer har blivit bättre, en trend som kallas ”the internet of things.” Molntjänster används ofta för att analysera massiva mängder data i och med dess enorma processorkapacitet, men är inte nödvändigtvis en fungerande lösning i avsides belägna områden med begränsad eller ingen internetkoppling. Edge-analys, d.v.s. att analysera data på platsen där den skapats, är ett möjligt alternativ till detta.

Edge-projektet är ett forskningsprojekt delvis finansierat med offentliga medel, med flera universitet och företag involverade. Detta diplomarbete baserar sig på arbete gjort under projektet, vars syfte var att undersöka edge-analys i motorinstallationer.

Målet med detta diplomarbete var att undersöka den teoretiska grunden av edge-analys och analysmetoder som användes under projektet. Dataflödet från signaler som kommer från sensorer, till datalagring, datarengöring, dataanalys och rapportering av data beskrevs. Relevanta verktyg, metoder och teknologier för dessa steg beskrevs även. Vissa koncept och metoder som är relevanta för analysarbete under projektet undersöktes. Vissa fall som skedde under projektets gång beskrevs, samt resultat från dessa.

Diplomarbetet beskrev problem som uppstod under projektets gång och problem som förväntas uppstå under ytterligare analys av data, samt möjliga lösningar till dessa problem. Ett förslag hur lägen av system kunde hanteras föreslogs.

De metoder som togs upp i diplomarbetet konstaterades ha potential i framtida arbete. Undersökning av läget i system konstaterades vara av särskilt intresse för framtida arbete.

**Nyckelord**: Edge analysis, cloud analysis, data analysis, edge installation, PCA, Markov processes, Monte Carlo, engine, machine learning, data pipeline

# ABSTRACT

In recent times, increasing processor power as well as more and new machine learning technology has led to increased interest in collecting and analyzing data, a trend which has come to be known as the internet of things (IoT). When it comes to massive amounts of data, a common approach is to process the data in cloud services with an enormous processing capacity. Cloud services are, however, not viable in remote areas, with limited access to the internet. Instead, edge computing, or analyzing the data on the site where the data is gathered, is a viable alternative.

The Edge project is a research project partially funded by public funds, with multiple universities and private firms involved. This thesis is based on work done during the Edge project, the purpose of which is to investigate the application of edge analysis in engine installations.

The objective of this thesis is to investigate the theoretical foundation of edge analysis and analysis methods used during the project. The flow of data from signals originating from sensors to data storages, data cleaning, data analysis, and reporting of data is described, as well as tools, methods and technologies used during those steps. Some methods and concepts relevant to the data and analysis worked on in the project are explored. The thesis then elaborates on a few analysis cases that appeared during the project, as well as on the result of those cases.

The thesis provides a description of both issues which appeared, and which may appear in the future during the analysis of the data. It also offers some possible solutions. A method of how state can be represented in systems was proposed.

The methods discussed in the theory part of this thesis work is likely to be useful in future projects. Considering how to represent and evaluate state is a topic which is concluded to be of particular interest in future work due to how it can simplify the way which analysis results are presented.

**Key words**: Edge analysis, cloud analysis, data analysis, edge installation, PCA, Markov processes, Monte Carlo, engine, machine learning, data pipeline

# TABLE OF CONTENTS

## Contents

# ACKNOWLEDGEMENTS

First, I like to acknowledge the project manager for the Edge project, Anders Öster, for his excellent leadership. The internal project team of Anders Korsbäck, Kim Westerlund, and Mikko Finell all shaped the journey of this thesis; the direction of this thesis would not have been the same without you. Kim Westerlund deserves a special mention for providing help and acting as a sounding board throughout the thesis project; thank you. In addition, I would like to thank Jonas Waller for his insights without which this thesis work might have taken a different direction, Tom Lillhonga for providing his expertise of PCA analysis, and Xiaoguo Storm for her expertise of the engine system. For good cooperation, I acknowledge Lauri Nyystilä and Andreas Lundell along with their respective teams, without the help of which this project would not have been possible.

# GLOSSARY

| | |
|---|---|
| Algorithm | A sequence of steps to take to achieve a computational task |
| Black box | A system whose inner workings are unknown. For instance, a proprietary computer system is most often a black box to an end user |
| CSV | Comma Separated Values, a text-based file format where data is separated by some sort of symbol, commonly commas. |
| Markov system | A system with the Markov property, meaning that if the current state of the system is known, future states of the system can be predicted |
| Matplotlib | A Python library made for creating plots and figures |
| Pandas | A Python library made for time-series data |
| PCA | Principal Component Analysis, a method where a set of data variables is summarized into a smaller number of variables whilst attempting to keep as much of the information in the original data as possible |
| Python | A high-level programming language commonly used for computer science tasks (among other things) |

# 1 INTRODUCTION

Edge computing at its core is to analyze data as close to its source as possible, i.e., at the edge. This typically refer to remote locations, such as a power station or an autonomous car, in contrast to a manned control room. Edge analysis offers some advantages and disadvantages compared to cloud-based solutions or older conventional approaches, which will be elaborated upon in this thesis.

The fundamentals of analysis remain largely the same despite which mode of analysis is used and analysis methods are highly dependent on the data and the goal of the analyses in question, whose theoretical background will be explored in the thesis.

## 1.1 Edge project

The Edge project is a research project partially funded by public funds, with multiple universities and private firms involved. This thesis is based on work done during the Edge project.

The project can be roughly divided into an initial phase with laboratory testing and measurement at the engine laboratory at VEBIC, and a second phase with measurement and testing at a real-life installation. This thesis focuses on the initial phase. Work was done in a small internal project team, as well as in cooperation with several other entities. This thesis describes work done on the project regarding data analysis and the detection of fault states.

## 1.2 Edge analysis and data transfer

Edge analysis has only relatively recently become more common, the main reason for which is the increase of use of sensors and other devices within the internet of things (IoT) creating massive amounts of data. Initially this data was often processed with cloud computing which has, however, created bottlenecks in communication latency and network bandwidth. According to a study (Premsankar, et al., 2018), at least certain demanding applications using the cloud and where low latency is required, need to use Edge computing to achieve low

latency.

## 1.3 Introduction to this thesis

This thesis investigates data analysis specific to the edge project, as well as data pipelines within edge analysis. Some specific cases from the project regarding data cleaning as well as exploratory analysis of data during the project is dissected. This thesis describes in part the work done as a part of the project in the form of different cases, and in part sets the theoretical background for the continuation of the work described in the thesis.

### 1.3.1 Data pipelines in edge analysis

This thesis will holistically investigate data pipelines within edge analysis, and the technologies, concepts and issues connected to these (see figure 1**)**. The initial step is to gather data from a data source to the edge device, in this thesis the source typically being a sensor. The specifics of how the data is transferred and parsed is not disseminated to any major extent in this thesis. The data is transferred from the edge device to cloud storage, where development of analysis methods occur, with the eventual goal of transferring those analysis methods to the edge. Before analysis, the data must be pre-processed by cleaning and wrangling the data according to requirements set by the analysis method. The workflow for exploratory analysis is explained. After analysis, the results are post-processed to validate results and assess uncertainty, which can then be reported.

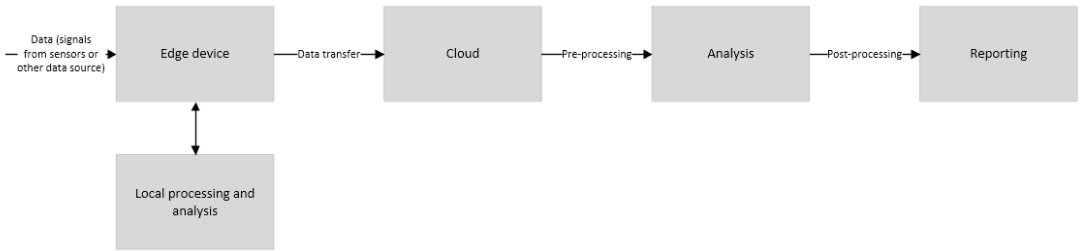


*Figure 1. A simplified flowchart of data pipelines in edge analysis as defined in this thesis. A more detailed flowchart of a possible real-world process is illustrated in figure 2.*

### 1.3.2 Data analysis

In this thesis many data analysis methods as well as related methods and concepts are explored. The purpose for this is to evaluate the use of these for the edge project, as well as to provide a theoretical foundation for other work done within the internal project team.

The primary data used in this thesis is a set of signal data from the VEBIC engine installation as well as data from accelerometers from the same installation. Some data cleaning and exploratory data analyses were done on these data sets in support of the internal project team.

# 2 THEORY

## 2.1 An overview of Edge systems

Edge analysis is, at least as a term, a relatively recent phenomenon. Although edge analysis in its most broad meaning, analysing data as close to the data source as possible, has existed for quite some time, Edge analysis is a term that contrasts with cloud analysis, and has increased in popularity for applications where cloud analysis is unsuitable.

### 2.1.1 Why edge analysis

There are a few reasons why edge computing (or making an edge device conduct analysis as close as possible to the source of the data) could be preferable to other methods. Firstly, internet might not be available, meaning that cloud computing is impossible. Secondly, if large amounts of data come from the sensors, this places higher requirements on the network. Cloud analysis also causes latency as data is transferred, analysed in the cloud, and then returned to device (if applicable), meaning that it might not be possible to conduct time-critical analyses on it. (Batewela, 2019)

Data in edge analysis comes from sensors, operational data from servers, as well as other possible origins points, and are often in the form of time-series data with relatively consistent sampling rates. A consistent sampling rate enables a selection of a device that is optimally dimensioned for the data flow, neither falling idle nor not being able to cope with the amount of data processing required. The form that these signals may take varies but is in edge analysis, eventually converges to an edge device, where the data processing occurs. For an example of data flow, see chapter 2.6.

### 2.1.2 Manned sensor analysis

In this context, a manned sensor is used to describe any kind of sensor setup which requires the intervention of people for the analysis. This could mean a laboratory environment, where the data is most often transferred automatically to on-site computers, but not fully automatically analysed. It could also mean a site, which is not typically manned, but whose

data is stored on-site and needs to be manually extracted by some sort of manual intervention.

### 2.1.3 Cloud computing

Cloud computing in a nutshell means connecting to a larger resource of servers, which can easily expand and contract according to computational needs (i.e., “the cloud”). In the context of sensor data analysis, data is most often transferred to and is analysed in the cloud; this has become so common that connecting sensors (as well as other devices) that IoT was coined to describe this trend.

There are a few characteristics that make cloud computing. Firstly, it is an on-demand self-service, meaning that the consumer can themselves acquire whatever resource they require. It is also easily accessible through standard mechanisms, meaning that almost any modern device with networking capabilities can make use of it. Resource pooling is utilized to serve many different customers using virtual servers, e.g., by using Docker Containers; this gives an economy of scale to cloud services with vast server capabilities even if each individual user only uses a fraction of the computing power of a server. This is combined with rapid elasticity, where computational resources of the desired kind can be rapidly deployed and decommissioned. The use of the resources is metered and can be reported to the consumer to provide transparency. (Jagidar, 2012) This also makes the most common monetization model a variation of paying only for what you use. The economy of scales involved in cloud services makes for efficient use of the service provider’s computational resources, which can make it more economical to run small-scale servers in the cloud rather than buying dedicated hardware.

Cloud services can be divided into three categories: Software-, Platform- and Infrastructure as a Service (abbreviated SaaS, PaaS and IaaS respectively). Software as a service can be seen as being made for an end user, in the context of data analysis, a data analysis tool. Platform as a Service is a step towards higher degree of freedom of configuration, providing an operative system and server stack network. This caters more towards application developers. Infrastructure as a service merely provides a server storage network, which gives the user full freedom to configure the system as they please, including operative systems. (Jagidar, 2012; Verma, 2018)

Cloud computing solves several of the problems that might occur with manned sensors. Firstly, it removes the need for manual transfer of data, as data is often automatically transferred to cloud storage, where it can be accessed from all over the world. This can create entirely unmanned sites of sensors. Secondly, it removes the need of buying discrete devices for data analysis, as analysis can happen in the cloud, or alternatively be downloaded and analysed on some other site. If extreme amounts of computational power are needed cloud computing is often a more affordable alternative than buying hardware for a single task if that hardware cannot be used continuously. Cloud storage also acts as a convenient centralized location for data, especially when comparing data, e.g., comparing similar data sources located in different parts of the world.

Another advantage to cloud computing is its ability to rapidly and efficiently increase or decrease however much computing power is necessary. For instance, a user can deploy or removes servers (typically with Docker Container or similar technology) where computation takes place; none of the calculations is done on the user's local device, which saves cost in buying hardware in exchange for costs for cloud services. This means a user can buy processing power rather than hardware itself, the latter which might not be cost-efficient if the user is unable to utilize said hardware enough. Cloud computing is also attractive in the case of massive amounts of data, where the only feasible way of processing it in any reasonable time is through massive computational power and tools such as Hadoop or Map Reduce, and acquiring the hardware needed is prohibitively expensive.

### 2.1.4 Edge computing

Edge analysis happens "at the edge", i.e., at the source where the data is produced. This is superficially similar to manned sensors but is typically done autonomously. There are several reasons why edge computing might be preferable to cloud computing, the most obvious being in locations where there is limited to no access to the internet. If real-time analysis is required, sending the data to a cloud service, analysing it in the cloud, and returning the result to the device might be too slow for the intended use of the analysis. It Is also possible that the amount of data being gathered is large and constant enough that using a discrete device to analyse it is cost-effective, but at the same time not so large that an inhibitive amount of computing power is needed to meet the needs. (Premsankar, et al., 2018)

One of the things lost when switching over to analysing at the edge as opposed to using cloud services is the ability to scale processing power according to needs. This means that if the data processing needs are sporadic and the hardware required expensive or otherwise impractical, edge computing might be an inefficient solution.

## 2.2 Handling dataflow

One of the major issues introduced with IoT is handling the flow of data. The increase in the number of sensors means an increase in the number of data sources, which means more data, and more types of data. On the analysis side, this means that more processing power is required. Cloud and edge analysis can help in the analysis process, as can optimizing working analysis solutions. However, in a broader sense, a larger issue is the fact that the demand of analysis work has greatly increased. In fact, when looking at how data correlates with each other, the amount of analysis that can be done increases exponentially, as every single type of data can in theory be compared to every other single type of data, in every possible combination. Two other issues can be found in the steps before and after data analysis: in the transfer of data to different devices, as well as reporting the data in a meaningful way.

### 2.2.1 Data transfer

An example of how a data analysis system could work is found in figure 2. Data always have a source, which in the case of this thesis are in the form of a set of signals gathered from sensors. These sensors send their signals to a gathering point, where the data conveyed through these signals are collected. In some cases, signals might be pre-processed by an edge device, especially when there is a large amount of data processing required before the data is in a usable form. In this example, data is collected in a database situated inside of an edge device.

An edge device can work completely alone depending on its purpose, collecting data, analysing the data in its database using some analysis algorithm and using the results to produce a decision through a decision-making algorithm. However, for a system to be able to work independently in a process where failure has large consequences, a high confidence in the system is required; it must provide accurate analyses and good decision-making without outside intervention, which is impossible to guarantee on an untested system.

To manage risk and build confidence in the system human involvement is necessary, and a convenient way to evaluate the analysis process is to send the data into a server from where researchers can collect them. In this example, a cloud storage system is used. In case of multiple similar installations, it is very convenient to collect all data in the same location to compare data from different installations to each other, and to the extent possible draw conclusions of commonalities in said data. Conclusions made in the cloud, either automatically by doing calculations not feasible to do at the edge, or manually by humans, can then be used to improve both the analysis and decision-making algorithms at the edge.

Once a decision-making algorithm produced a decision, the analysis system then acts on said decision. In testing, as confidence in the decision-making algorithm is lacking, the decision could be what status to display, or whether an alert or recommendation should be displayed to relevant personnel. As confidence in the accuracy of predictions increases, the system can be given a higher degree of autonomy. A system such as this could but might not necessarily have the goal of eventually working entirely autonomously by removing the cloud-computing component entirely.

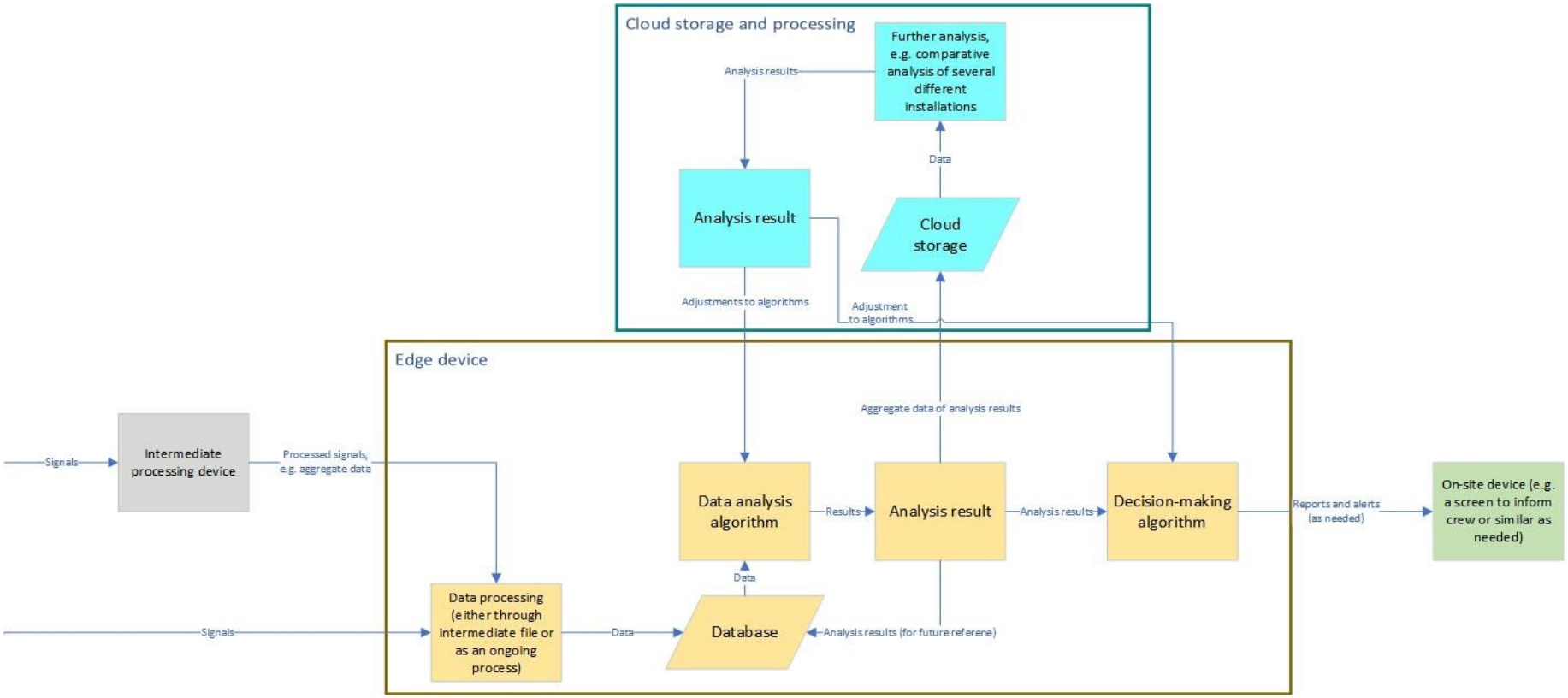


*Figure 2. A diagram of dataflow in a hypothetical data collection and analysis system.*

### 2.2.2 Reporting data

Typically, the results of analyses are worth very little unless fit for human consumption to enable decision-makers to make better decisions. Considering that deep technical analysis is time consuming and only in the realm of experts, it is important that data should be reported in a comprehensible and useful way. The metrics employed should in other words be informative in that it tells something useful about the state of the system and be presented in a way that the end user can understand, and which is actionable.

The term data reporting is often used in the business world as a step where data is visualized, and analysis is made based on those reports. In the data reporting pipeline as proposed earlier, reporting is done as a final step and product to the end user. Although simple metrics can be used, the report may also display some black box predictions and/or metrics to the end user.

In an initial, experimental stage, when the system enters new states, which may or may not indicate some sort of fault, there will be an issue of being uncertain of what kind of recommendation should be made – there are many faulty states a system can be in, but there can also be more than one state, where the system functions as intended. One way to handle it could be to display a warning that the system has entered a state, about which it has insufficient data, ideally with some sort of qualification of what is new about that state. That way the end user can use their own judgement in cases, where the system itself does not yet have confidence in making predictions, as opposed to cases, where the system can report with confidence when it has detected an error state. Then, ideally, with time, the system can make better and better predictions of its own of what sort of states can signify some type of fault, and what states are fine. For more on states, see chapter 2.6.

### 2.2.3 Data analysis

Data analysis can be defined as the process of making sense of any given set of data by cleaning, transforming, and modelling it. After some sort of understanding of the data have been gained, the analysis results can be used to make general conclusions, or to support decision-making. Data analysis can make use of any number of different methods, commonly

using metrics and data visualization. When working on unfamiliar data, exploratory data analysis is done to gain a basic idea of the data, as well as formulating some hypotheses regarding the data, which can then be tested. There is no real upper limit to how much exploratory data analysis can be done on a set of data, though the goal of exploratory data analysis is often to make useful discoveries about the data.

In the data pipeline as proposed earlier analysis will be done continuously on well-known data, which means that using a set of standard methods to calculate metrics will likely perform well in most circumstances. A machine learning model could be used as well, which, if configured to, can learn and change as it receives new data. See chapter 2.10 for more information regarding machine learning.

## 2.3 Data processing

Data as received from the source is oftentimes not fit for use in any given analysis as-is. Common problems are that the data is not in the correct form to easily make the analysis required, or that there are errors or irrelevant data in the dataset. Failure to correct these kinds of errors can result in faulty conclusions being drawn from the data, or conclusions that could have been made with processed data never becoming clear with unprocessed data. Furthermore, good data saves time during the analysis phase, both for human analysts and machine learning algorithms. (EliteDataScience.com, 2016)

Not all errors have to be (or can be) corrected – sometimes measurement errors or some other form of error is large enough in magnitude that a given error caused by faults in the data is negligible. Instead, it makes sense to focus on the errors that cause issues for the analysis methods employed.

### 2.3.1 Data cleaning

Data cleaning is about removing faulty or irrelevant data from the dataset. There can be any number of different ways data can be faulty or irrelevant in respect towards an analysis method, but there are a few common categories that appear in many different problems.

Duplicate entries appear surprisingly often in datasets, typically due to an error in combining data. These can simply be removed.

Structural errors are when data is inconsistent with tags or categories, which splits data that should be in a single category in several categories. These can simply be consolidated into a single category.

Sometimes, but not always, outliers should be filtered out, if a sufficient reason can be given for why it should be dropped. Similarly, when it comes to data points where one or more variables are missing data, one way to handle it is to drop it from the data set, but alternatively it is possible to replace the missing values if the true value is known with reasonable certainty. As always, it is up to the analyst to use their knowledge of the analysis method, the data, and their experience to determine what can be done without creating misleading results. For analysis methods using machine learning, it is recommended that the machine learning process should be told of the missing data in some way. (EliteDataScience.com, 2016)

Aside from these common ways of cleaning data it is also important to check whether the requirements that needs to be fulfilled for the data analysis to be valid are fulfilled. This could mean doing exploratory data analysis or tests on new datasets even if they are from the same source as previously explored datasets, as the new datasets can have new faults or other changes in the data that the old dataset did not. It is especially important to validate data if it is used in an automated system, where the system itself might be unable to catch otherwise obvious errors, which can lead to the system making incorrect decisions.

A widely applicable metric for ensuring that data quality requirements are fulfilled for sensor data which is sampled at regular intervals is data integrity, which Zhao et al (2017) defines as follows:

$$Data\ integrity = \frac{N_{real} - N_{missed} - N_{invalid}}{N_{real}} \quad \textbf{(1)}$$

Where $N_{real}$ is the number of measurement points expected to be received for a given time period, $N_{missed}$ is the number of measurement points missing from that time period, and $N_{invalid}$ is the number of measurement points that contain errors (which are typically discarded for analysis purposes). Data integrity is a percentage of accurate data points for a given period where, if the percentage is too low, it is unlikely that the data is good enough to

be used as evidence to draw conclusions. It should also be noted that partitioning data in time periods smaller than the entirety of the data is typically preferable so that short bursts of disturbances is detected.

### 2.3.2 Data transformation and aggregation

Given that there are infinite different ways analysis of a dataset could be made and that data typically is only saved in one way, it is inevitable that transforming data to suit the need for different forms of analysis is inevitable. If it is not obvious what parts of the data is going to be useful it is often preferred (if practical) to save the data in the most granular way possible. This means saving every single observation taken in each period. A lot of data analysis summarize data in different ways to draw conclusions of the data. One way to summarize is to resample data, which groups different observations together for instance either by category or time periods and aggregate the value in these observations in some way, e.g., by summing up the values or averaging them.

Another reason why granular data is often preferred is that they contain as much of the original data as possible. Aggregating data invariably leads to a loss of data; it is impossible to determine what the original values of two or more rows were from the average of all those values alone. During exploratory analysis and in cases where memory use and calculation times are low, a good workflow is to copy the original data and process it through the entire data analysis process from start to end, rather than working on an intermediate step in the data. This way changes made to previous steps in the process is always reflected in the final result.

## 2.4 IT systems and automation systems

When comparing automation and IT-systems, IT-systems tend to be less critical; failure typically means temporary outages for customers in mostly non-critical applications. Automation systems on the other handles physical systems – e.g., a factory, or a motor, etc. These systems need to be maintained within certain limits to stay safe, for instance, a robotic arm which flings a heavy object could damage itself, objects within its environment, or injure or even kill people.

Automation systems tend to use lower-level programming languages (seeing that

Programmable Logic Controllers tend to have less memory than computer hardware) which does help in limiting the number of parts where errors can occur within an automation system as compared to more general IT software (which often uses any number of different libraries and modules). This is especially apparent in the cases where ladder diagram programming is used, where programming is almost entirely limited to basic logic gates.

### 2.4.1 Scope, requirements and testing

Creating completely bug-free code becomes practically impossible after a certain degree of complexity. Not only does the program have to contend with the complexity within a program itself, but also any potential bugs in any code libraries it uses, the operative systems it uses, as well as any other program which interacts with the program. This is a problem for critical systems, i.e., systems where failure leads to severe or even catastrophic consequences. Seeing that making bug-free code is impossible, setting a flawless program as a goal is doomed to fail. Instead, in a critical system, a method to guide the goal is to set requirements according to what the system needs to be able to achieve to stay safe in every conceivable situation which can be reasonably be handled. This, along with the goals, should help pinpoint the scope of the system. See figure 3 for a possible way to achieve this. (Englund, 2018)

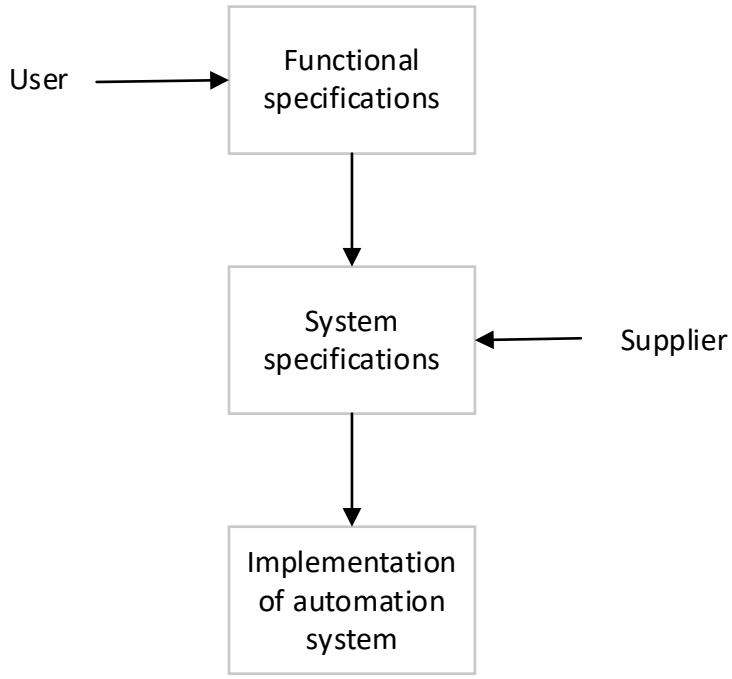


*Figure 3. Specification and implementation of an automation system. (Englund, 2018)*

On the programming side requirements could be that the system must be able to keep on running critical processes, or alternatively shut critical processes down safely should an error occur. It needs to be robust enough to handle unexpected input values and situations. Within software development a commonly used tool to ensure that requirements set are adhered to through every step of development is test-driven development, where the basic requirements of a program is defined as a set of tests. When the tests pass, the program is done in the sense that the pure minimum of features required are fulfilled, but often the code and tests are improved upon (refactored) at this point. (Janzen & Saiedian, 2005) A similar concept is continuous integration, where testing is done within an automated pipeline with the purpose of ensuring that bugs are not introduced before being put to practice. (Fowler, 2006) Of course, the complexity involved in software means that in practice it can be difficult and expensive to predict every conceivable situation and create tests cases for each one of them.

From the perspective of automation systems, testing is about ensuring that the logic within the automation system works as intended. Unintended behavior could be that the system does not respond properly in possible situations which were overlooked or in edge cases, but it is also possible that the system does not act as intended even in situations the system was designed for. A common testing tool for automation systems is simulation, where different situations can be safely designed and simulated, both normal situations as well as edge cases. An example of a tool used for simulating automation systems is Simulink. (Lobo, 2018)

## 2.5 Equipment degradation

Over time equipment degrades, whether from mechanical wear, chemical reactions etc. In the short-term degradation leads to steadily decreased performance, but if left unchecked faults in both sensors and machines can lead to failure with potentially catastrophic consequences. The key is to ensure failures never happen by employing e.g., planned maintenance or through condition-based maintenance.

### 2.5.1 Machine faults

A fault is defined by Olsson (2004) as “an abnormal state of a machine or a system such as dysfunction or malfunction of a part, an assembly, or the whole system.” Machine faults vary

in severity and might or might not be detectable. According to Nakamura (2007), machine faults start with a gradual slope, increasing in severity over time, although the onset of the fault can be different (compare an axle snapping to a bearing getting worn).

### 2.5.2 Sensor drift

Ideally a sensor should produce the same measurement every time, if what it measures stays at the same level (for instance, a temperature sensor measuring an unchanged temperature twice). Over time the output of the sensors for a given level changes, a phenomenon called sensor drift.

Take for example a thermocouple, which uses the thermoelectric properties of two different materials (typically metals) to produce a voltage, and in turn using that voltage to determine the temperature. One of the main reasons behind sensor drift in a type K thermocouple is “green rot”, which is when chromium oxidizes, changing the composition of the materials in the thermocouple, which changes its thermoelectric properties. There are many causes of sensor drift (which of them being the most significant depending on the technology and method used in the sensor in question, as well as the environment the sensor is in), though two common causes are metallurgical changes and chemical changes over time. Mechanical damage or water damage are two other common causes of false reading in sensors. Even for the same model of sensors natural variations in the composition of the materials used exist, which affects the expected progression of sensor degradation. (Schuh, 2003)

The typical solution to sensor drift is to either recalibrate sensors, or use known good reference measurement to compensate for drift. Recalibrating sensors is not desired in unmanned installations and depending on the sensor involved might be expensive and time consuming. For unmanned installations, one solution could be to have redundant sensors at the same spot, which shows if a sensor starts to drift wildly from the values of the rest of the sensors. Given that there are at least three sensors, if a single sensor would drift as the others remain the same, the sensor which drifted from can assumed to be faulty and its measurement values disregarded, as well as possibly being flagged for replacement. Using multiple sensors does also increase both costs, as well as the complexity of the system. Furthermore, using multiple sensors does not detect the absolute values of sensor drift in the case drift happens consistently between every sensor used. However, if the expected rate of deterioration for the relevant environment the sensor is used in is known that might be used to make an estimate of how

large the sensor drift should be.

Another theoretically possible way to detect when a sensor measurement is drifting is to compare the output of the sensor measurement with other, tightly connected, measurements. For instance, given that output temperature of exhaust gas, the amount of energy in the fuel consumed, the volume, pressure, and composition of the air, and that any other energy streams away from the system are all known, it is possible to calculate what the output temperature should be, and then compare that to what the output temperature is. If the calculated value diverges from the measured value, this indicates that a sensor could be faulty, although not necessarily which sensor. It might also indicate another issue with the system itself, i.e., if a leak would create an additional energy stream away from the system which is not considered in the calculations.

## 2.6 Markov systems and models

A Markov system is a system for which future states can be predicted with information about its current state alone. Markov models can be divided into being autonomous or controlled, and fully observable or partially observable.

### 2.6.1 Fully observable Markov models

The simplest case, of an autonomous model with a fully observable state, is modeled with Markov chains. Markov chains are essentially a chain of different states which has certain probabilities to enter another state from the state which they currently are in. See figure 4 for an example of a Markov chain.


*Figure 4. A picture of a Markov chain. In this example system, if it is not windy, there is a 70% chance of it remaining now windy and a 30% chance of it becoming windy. (Soni, 2018)*

When a fully observable system is controlled, it becomes a Markov decision process. The difference is that a Markov decision process has an agent which makes decisions that affects the future state of the system, see figure 5 for an example of a Markov decision process.

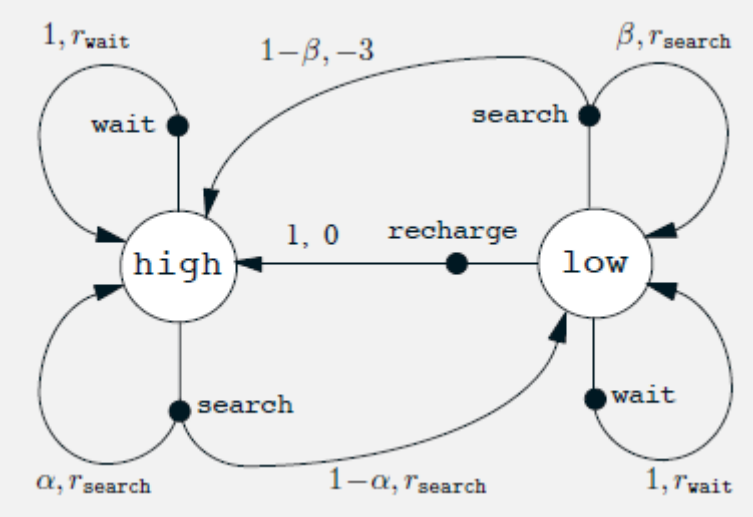


*Figure 5. An example of a Markov decision process, of an agent (a robot) with the goal of recycling cans in an office. It has two states, high and low power, and three actions, waiting to receive cans, searching for cans, and (if it has low power) going to recharge. The probability of the next state depends on what action is taken in what state. (Sutton & Barto, 2018)*

### 2.6.2 Markov state and Markov system

A state is a set of variables describing a system at a given point in time. Variables can be continuous (e.g., temperature of cooling water, amount of power generated) or discrete (any Boolean variables, but also any other variable with a finite number of values such as if the load of an engine is increased/decreased/held steady).

A Markov state contains enough information in its variables that it possible to use it as a complete summary of the past, which in turn can be used to predict future states. A system which, in its state, contains enough information that it can be used to predict future observations is said to have the Markov property. Put in another way, it does not matter in what manner a system reached a particular state, the information in its current state is enough to predict the next state, and since the next state of that state can also be predicted and so on, the entire future of the system can be predicted. If some piece of information unavailable from the current variables is needed to achieve a Markov state in an otherwise informationally

sufficient state (e.g., when a load change happened), but is available in the history, it is possible to store that piece of information as variable to achieve a Markov state. (Sutton & Barto, 2018)

### 2.6.3 Agents and Markov Decision Process

An agent is an entity able to observe the state of a system, as well as take actions to affect the system. In the case of an agent acting on a Markov system, the agent state is the agent's approximation to a Markov state – it does not necessarily have enough information to be able to predict anything completely accurately. The agent takes the information it has in its state in the form of variables, and inputs it into a function to determine what action to take, as determined by what goals it has. A system with the Markov property which has an agent is called a Markov Decision Process (MDP).

The goal of the agent can be stated as maximizing the output of a reward signal over a certain time frame, which is a number rewarded to the agent for some certain state or event. The reward signal comes from a function of differing complexity, from a simple sensor readout used directly to a function taking any number of different things into consideration. Reward signals such as for "fuel efficiency" can be determined quite easily, whilst a reward signal for "happy customers", is difficult to formulate; on the other hand, if the goal was to have happy customers and fuel efficiency was chosen as an approximation for that goal, the agent might not necessarily act in a way that actually produces happy customers, as customers might not care about fuel efficiency, or some other aspect important to them might be ignored by the agent.

As in an MDP, the action not only affects the next state, but also all states to come, the agent can make decisions not only dependent on what action would return the best reward in the next state, but also predict what reward taking a certain action would lead to in the future. (Sutton & Barto, 2018)

### 2.6.4 Partially observable Markov models

When the system is partially observable and autonomous, hidden Markov models (HMM) are used. One can regard an HMM as being an augmentation of a Markov chain for systems when

some information regarding the system is hidden.

Ramage (2007) describes the three different problems that might be asked of HMMs; likelihood (given the likelihoods of that the initial state is a certain state, the transition likelihoods of states, and the observation likelihood, that is, the likelihood that a value is generated from a given state, and a set of observations, determine the likelihood of a given observation is in a given state), decoding (given an observation sequence as well as the initial state likelihoods, transition state likelihoods, and the likelihood that a value is generated from a given state, discover the best hidden state sequence), and learning (given an observation sequence and the set of states in the HMM, learn the likelihood of transition between states and the likelihood of a state giving out a certain value).

The final option, of a partially observed system being controlled, is a partially observable Markov decision process. Other than the difficulties posed by a partially observed system, the agent itself also needs keep track of the state it is in. However, as the agent does not know for certain what state it is currently in, an agent in a partially observable Markov decision process also contains a belief state. As the system is assumed to be Markovian, this belief state is based wholly on the previous belief state, the action taken, and the current observation.

Spaan (2012) identifies two different reasons why the state would only be partially observable, the first being that sensor readings are noisy in that the same state can result in many different sensor readings, the second one being that multiple states give the same sensor reading; that is, only a limited part of the environment can be sensed.

Two or more states that look the same from sensor readings, but are two discrete states, forms what in this thesis will be referred to as a superstate, the expected outcomes of which is dependent on the relative sizes of the different actual states that make up the superstates' population, as well as the expected outcomes of each of those states. This can trivially be described according to (2).

$$T_{S_a} = \sum_{i=0}^{n} T_{s_0} * p_{s_0} + T_{s_1} * p_{s_1} + \cdots T_{s_n} * p_{s_n} \qquad \textbf{(2)}$$

where $T_{S_a}$ is the transition probabilities between the apparent state, and $T_{s_0}$ is the transition probability of the first state, $p_{s_0}$ the probability of the first state being the true state when the apparent state is observed, and so on for all n states that make up the apparent state.

One thing to note is that the relative population of the states which look identical to each other are probably highly dependent on the previous state and the action made, meaning that it is likely that different expected result could come from that same superstate. In other words, if you would start running an engine in a completely different way or in a completely different environment, the relative population of a superstate would likely change, as would the expected outcomes. This would lead to worse predictions. This can be partially solved by using methods for partially observed Markov systems, or by observing enough of the environment that the state is fully observed (if applicable).

## 2.7 Monte Carlo method

According to Sutton & Barto (2018) Monte Carlo is a term that is often used for any calculation method which uses a significant random component to estimate the result. The underlying principle that Monte Carlo methods has in common is that as more samples are gathered, the average of these samples eventually converges towards the expected value.

The Monte Carlo method can be seen as a form of numerical integration. Suppose that a quantity of interest x with the distribution *h(x)* is needed. If *h(x)* is known, numerical integration could be used, but often it is unknown. If the underlying processes are known, however, one could simulate those underlying processes using Monte Carlo, and get the expected value of *h(x)*, or even the distribution of *h(x)*. (Dunn, 2011)

### 2.7.1 Basis of Monte Carlo

Monte Carlo is based on using stochastic variables, or random variables. The values of stochastic variables cannot be specified in advance of those values being observed. For example, the maximum temperature can be predicted with a reasonable amount of certainty for a given day, but not precisely. The time the sun rises any given day, however, can be predicted accurately, and is therefore not random.

Furthermore, Monte Carlo also uses the law of large numbers and the central limit theorem. The law of large numbers state that, if the mean exists and the variance is bounded, the average

of the samples taken of the distribution equals the actual mean as the number of samples approaches infinity. This can be formulated in the following expression

$$\overline{z} = \frac{1}{N}\sum_{i=1}^{N} z(x_i) \quad \textbf{(3)}$$

where $N$ is the number of samples, $\overline{z}$ is the estimated expected value, $z$ is a function of $x$, and $x_i$ is sampled from the probability distribution function $f(x)$. When the number of samples reaches infinity, $\overline{z}$ will equal the population mean $\langle z \rangle$, that is

$$\lim_{N\to\infty} \overline{z} = \langle z \rangle \quad \textbf{(4)}$$

(William L. Dunn, 2011)

### 2.7.2 Single random variables

There are two kinds of random variables, continuous random variables, and discrete random variables.

The probability density function (PDF) of a random variable has three properties. Firstly, it is defined on an interval *[a, b]*, where *b* is greater than *a*, secondly, it is non-negative within that interval, and thirdly, it is normalized so that

$$\int_a^b f(x)dx = 1 \quad \textbf{(5)}$$

or alternatively formulated, the area of the function within the interval *[a, b]* equals 1. The interval can be either finite or infinite.

The PDF specifies probability per unit of *x*. For a continuous variable, the probability of receiving any single value is 0 as the output can be an infinite number of values. Instead, the probability of a value $x_i$ being within some interval of the PDF is calculated. This can be stated in the form

$$f(x)dx = Prob\{x \leq x_i \leq x + dx\} \quad \textbf{(6)}$$

Two common distributions for continuous variables are the uniform distribution as well as the Gaussian distribution, where the PDF for the uniform distribution is

$$f(x) = \frac{1}{b-a} \quad \textbf{(7)}$$

and for the Gaussian distribution is

$$F(x) = f(x|\mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right] \quad \textbf{(8)}$$

where σ is the standard deviation and μ is the mean value.

To calculate the probability that a given variable is equal to or smaller than some number, the cumulative distribution function (CDF) is used. The CDF is defined by

$$F(x) = \int_a^x f(x')dx' \quad \textbf{(9)}$$

Where $f(x)$ is the PDF over the interval *[a, b]*. To calculate that a value is within two variables the following formula is used

$$Prob\{x_1 \leq x \leq x_2\} = \int_{x_1}^{x_2} f(x)dx = F(x_2) - F(x_1) \quad \textbf{(10)}$$

A discrete random variable is only defined for discrete values, for example, a normal six-sided die only produces discrete values from 1 to 6, which is notated as $x_1 = 1$, $x_2 = 2$, …, $x_6 = 6$. The number of outcomes for a discrete random variable can be finite or unbounded, the requirements for discrete random variables being that $_i \geq 0$ and

$$\sum_{i=1}^{I} f_i = 1 \quad \textbf{(11)}$$

or that the sum of all values within the PDF for a discrete variable equals 1.

The CDF for the discrete case is the probability of one of it being one of the first *i* events and is defined as

$$Fi \equiv \sum_{j=1}^{i} f_j \quad \textbf{(12)}$$

(Dunn, 2011)

### 2.7.3 Central limit theorem

With the help of the central limit theorem, it is possible to get an estimate of the uncertainty of any expected value obtained by the Monte Carlo method. Any uncertainty in the estimated expected value is proportional to $1/\sqrt{N}$, where N is the number of samples or histories of *f(x)*. For instance, this means that if the number of samples is quadrupled, the uncertainty of the estimate of mean is halved.

To estimate the uncertainty of a mean value estimated from Monte Carlo with respect to the true mean, the following formula can be used

$$Prob\{\overline{z} - \lambda s(x)/\sqrt{N} \leq \langle z \rangle \leq \overline{z} + \lambda s(x)/\sqrt{N}\} \simeq \frac{1}{\sqrt{2\pi}} \int_{-\lambda}^{\lambda} e^{-u^2/2} du \qquad \textbf{(13)}$$

where $\overline{z}$ and $\langle z \rangle$ is the estimated expected value and population mean respectively, $\lambda$ is the standard normal deviation, $s(x)$ is the calculated standard deviation of sample history x, $e$ is Euler's number, and $u$ is the population mean.

The probability for a value to be placed in certain value of lambda can be numerically calculated with the expression on the right side of the above formula. The right side of the formula is the normal standard distribution integrated from $-\lambda$ to $\lambda$. (Dunn, 2011)

### 2.7.4 Monte Carlo Simulation

Many problems can be formulated in the form of

$$\langle z \rangle = \int_a^b z(x) f(x) dx \qquad \textbf{(14)}$$

Where the PDF $f(x)$ is not known beforehand. However, in many of these cases it is possible to simulate the PDF and estimate an average value. In these simulations, reality is simulated

as closely as possible, often by use of probability models. Monte Carlo Simulations which mimic reality numerically are called analog Monte Carlo Simulations.

Analog Monte Carlo simulations often require a lot of computation power as a potentially complex reality needs to be at least sufficiently well simulated, or a very large number of some form of event needs to be simulated. It is still possible to get around this by creating biases in the simulation, say if you wanted to simulate particles in orbit hitting a satellite, you could force those particles to move towards the satellite, and not in any other direction. Adding a bias alone would create an inaccurate simulation, the key instead being that after a result is calculated with the bias, the bias is then corrected to undo it, i.e., by re-adding all the particles that were not flying towards the satellite and taking the average from that. The end result is that you get the result you are interested in whilst simulating many fewer particles. Monte Carlo simulation using biases is called nonanalog Monte Carlo simulations. (Dunn, 2011)

## 2.8 ROC curve

Receiver Operator Characteristic (ROC) curve was originally developed during World War II as a standardized system to evaluate a signal receiver's ability to identify objects of interest correctly from background noise. (Janet E Joy, 2005) ROC curves proved to be a useful tool which since then has been adopted within many different fields to evaluate the accuracy of tests or models, the medicinal field being a notable adopter, with Metz describing the basics of ROC curve for a medical audience as early as 1978.

ROC curves are used to assess classification models, typically binary true or false. The ROC curve requires a value which represents how certain a model or test is on the fact that the a given observation is positive (with lower certainty implying that a given instance is negative) as well as labels signifying whether the row was positive or negative in reality.

### 2.8.1 Accuracy

The accuracy of a test or a model can be defined as the number of time that test or model is correct in its decision. Accuracy does not provide a full picture; take for example a disease that only affects 1% of the population. Creating a model which assumes that no one has the

disease will have a 99% accuracy rate but will also incorrectly identify 100% of the people who has the disease as not having it. In cases where a low percentage outcome has catastrophic consequences it can be more important to correctly identify the cases where the catastrophic outcome can happen, even if it means more false alarms. Two measures that can give a more accurate picture of the model is sensitivity and specificity.

Sensitivity is defined as

$$Sensitivity = \frac{Number\ of\ true\ positive\ decisions}{Number\ of\ actually\ positive\ cases} \quad \textbf{(15)}$$

Where true positives are cases in which the test or model accurately decided that a case was true. A false positive is when the test or model incorrectly decided a case was true.

Specificity is defined as

$$Specificity = \frac{Number\ of\ true\ negative\ decisions}{Number\ of\ actually\ negative\ cases} \quad \textbf{(16)}$$

Where true negatives are cases in which the test or model accurately decided that a case was negative. A false negative is when the test or model incorrectly decided a case was false.

The true number of positive and negative cases must always add up to 100% of the cases regardless of if the model decided they were positive of negative, which gives us the following two formulas

$$TPF + FNF = 1 \quad \textbf{(17)}$$

$$TNF + FPF = 1 \quad \textbf{(18)}$$

For completeness, accuracy is defined as

$$Accuracy = \frac{No.\ correct\ decisions}{No.of\ cases} = \frac{No.\ true\ positive\ decisions}{No.of\ cases} + \frac{No.\ true\ negative\ decisions}{No.of\ cases} \quad \textbf{(19)}$$

Sensitivity is also called true positive fraction (TPF) and specificity true negative fraction (TNF). (Metz, 1978)

When fine-tuning a model, the consequences of all four possible outcomes (true positives, true negatives, false positives, and false negatives) must be considered to tailor the model to the real-world application optimally.

### 2.8.2 Decision threshold and the ROC curve

An example of the output of positive and negative cases can be distributed across a test or model's output value, and how it affects the rate of wrong decisions can be seen in figure 6.

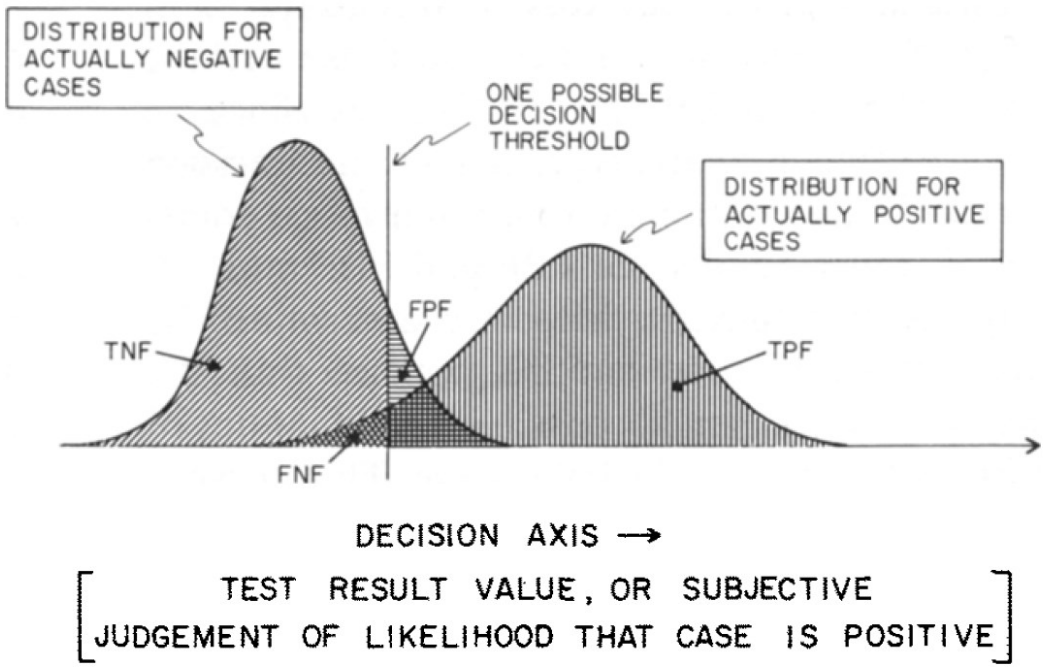


*Figure 6. A hypothetical distribution of the value used to make a decision, and its distribution of actually true and false cases. (Metz, 1978)*

On its own, distributions like these does not necessarily show where to place the decision threshold. Instead, ROC curves can be utilized to show how changing the decision threshold changes the rate of true and false positives, see figure 7.

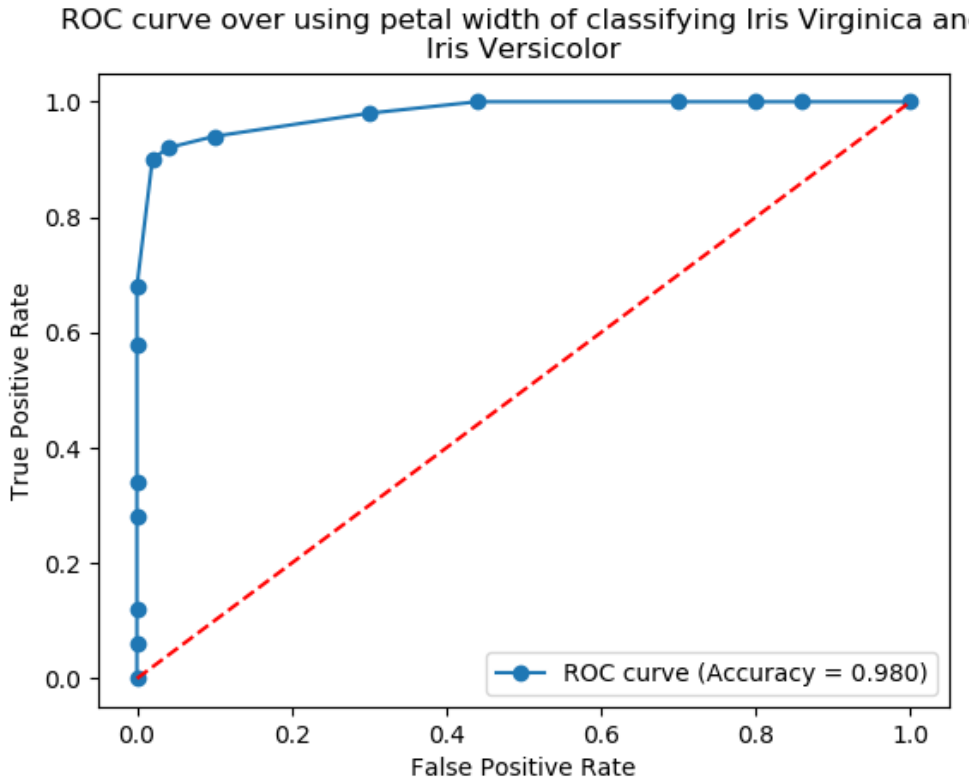


*Figure 7. Example ROC curve using the petal width of two flowers as a classifier. The red line represents the baseline, which would be no better than a random guess. (Fisher, 1936)*

The ROC curve plots the false positive fraction on the x axis and the true positive fraction on the y axis. The curve will inevitably go through the lower left corner (as all tests can be labeled as negative) as well as the upper right corner (as all tests can be labeled as positive). Plotting the negative fraction is unnecessary as the true and false negative fraction can be calculated from the false and true positive fraction respectively, according to (15) and (16). With those things kept in mind, in a good model, the plot will sharply move to the area around the upper left corner, and in a bad model, the plot will move along the center of the chart (as good as random guessing). A model which moves underneath the center (worse than random guessing) can have its result be reversed to achieve a success rate of over 50%. Also, it is possible to see if the theoretical grounds that the test is based on is sound if the slope of the ROC curve steadily decreases as one moves to the upper right corner of the curve, as this implies a tail of outliers which slowly move towards zero negative cases. It is possible to calculate the accuracy of the test or model from a ROC curve by calculating the area under the curve; if the area is 0.90 for instance, the accuracy of the test is 90%.

As ROC curves compare the true and false positive fraction it gives a more representative picture of the merits of the test or model in cases where the population is skewed to either

negative or positive cases.

Once a model or test has been determined to be useful, the next question is where to put the decision threshold. One way is to calculate the net benefit of testing through the following formula.

$$\overline{NB} = \{(C_{FN} - C_{TP}) \times P(D+)\} \times P(T+|D+) - \{(C_{FP} - C_{TN}) \times P(D-)\} \times P(T+|D-) - C_0 \quad \textbf{(20)}$$

Where $P(D-)$, $P(D+)$ is the probability that a case from the studied population is actually negative and actually positive respectively, $P(T+|D+)$ and $P(T+|D-)$ is the probability that an actually positive case is diagnosed to be positive and an actually positive case is diagnosed to be negative respectively, $C_{TP}$, $C_{FN}$, $C_{TN}$ and $C_{FP}$ is the cost of a true positive, false negative, true negative and false positive decision respectively, and $C_0$ is the cost of conducting a test. (Metz, 1978)

For a setup where the sensor analysis system is continuously doing tests, the net benefit could be seen as the net benefit of all false positives and true positives throughout the lifespan of the sensors, with the cost of all those tests being the cost of buying/installing/maintaining the sensors (assuming operating the sensors has negligible electricity costs, and there are no other additional costs of measuring with the sensors frequently). If the sensors are precise and the test in question is relatively insensitive to the momentary values of natural variation in the process, an increased number of tests within the same timespan should not create false positives. If false positives are an issue due to an increased testing rate, a possible solution would be to filter the results so that a certain density of positives over are required to be considered a positive by the system overall, which could probably be a decent method to employ if the test detects a positive by looking for a long-term state change. Methods such as these are of course entirely dependent on the test and the cause of the wrong predictions of that test; for instance, if the fault will get more obvious over time and the confidence of the test will likewise increase or will alternatively remain unchanged if the degree of the fault does not factor in as significantly in producing a correct or wrong prediction. An exception would be if a discrete precursor event that is not noticeable afterwards is to be detected, for instance, if the test tries to detect a fault caused by a bolt becoming loose and falling off, but this fault is not itself detectable afterwards. This sort of errors would be more difficult to verify through any of the above-mentioned methods. However, it is possible to try and avoid tests that rely on this sort of properties in favor of other form of tests.

To get the net value over a complete life cycle, one possibility would be to use the Monte Carlo method. As an example of how this could be done, take this modified expression from above where the time-dimension is added.

$$\overline{NB}(t) = \{(C_{FN} - C_{TP}) \times f_{P(D+)}(t)\} \times P(T+|D+) - \{(C_{FP} - C_{TN}) \times f_{P(D-)}(t)\} \times P(T+|D-) \quad \textbf{(21)}$$

Where $\overline{NB}(t)$ is the net benefit of taking a test at time *t* and $f_{P(D+)}(t) + f_{P(D-)}(t) = 1$ for $t \geq 0$. $C_0$, which in this case would be the cost of installing and maintaining the system over its lifespan, would have to be considered separately, most likely by adding it to costs. If the costs involved in failure of the test, or accuracy of the test changes over time, it is possible to also make them functions dependent on time as well. The Monte Carlo method in question could, for instance, simulate the system many times by randomly generating the state of the machine for a chosen discrete time step according to the population function (by integrating over said time step), and taking the consequence of said step. The cost of a true negative would be profit from "business as usual", whilst any other result would involve some sort of additional action, which would incur some sort of cost on the normal "business as usual" state, as well as possibly some sort of opportunity cost, depending on the nature of said action. Obviously, a breakdown would in almost all cases be significantly more expensive than preventive maintenance and repair.

As faults that have been recently corrected are unlikely to appear again, any maintenance actions or repairs connected to said fault should probably influence the population functions in the simulations, for instance by setting a new $t = 0$ for the population functions at the time maintenance occurred, assuming that is an accurate enough approximation. It is of course possible to add any number of additional factors into consideration in the model, depending on need. By running such a simulation repeatedly, it is possible to get an approximation of the value in question – see chapter 2.7 for more information on the basis of the Monte Carlo method.

An issue would be that the functions of the populations of actually positive and negative cases are in many cases not trivial to create due to the amount of data and research required, seeing that these are almost exclusively based on empirical studies. This is compounded if further factors are wanted to be taken into consideration, such as how the system is used, the

environment the system is in, maintenance, etc. etc.

## 2.9 Principal component analysis (PCA)

Principal component analysis is, at its core, a dimensionality-reduction method. For instance, if a data set with an arbitrary number of measurement points and 100 variables, principal component analysis can be used to reduce the number of variables involved by combining variables into principal components; the number of data points is the same, but the number of variables (or dimensions) is reduced. Combining two or more variables into a principal component will, as all reduction type operations, result in data being lost, but principal component analysis seeks to reduce the amount of information lost during this process. This also means a reduction in the amount of redundant data.

The method of reduction of two variables into a single principal component can be visualized by plotting the two variables against each other, see figure 8**.** The line which makes up the new principal component is whatever line which maximizes the variance (or distance) of the maximum and minimum point, where the points used are the closest point of the line from the original points. The principal components are determined using Singular Value Decomposition (SVD), which will not be elaborated upon in this thesis.

To figure out the optimal line (new axes) for the data, take the standardized points of the data, and make a line (projection) to the closest point on the line. To preserve as much of the original information as possible, these projections that point to the closest point on the line should have as large a range as possible to preserve as much data as possible, or in other words, the projections should have as large a variance as possible. With more than 2 dimensions, the principal component makes up a new plane. (Jaadi, 2019)

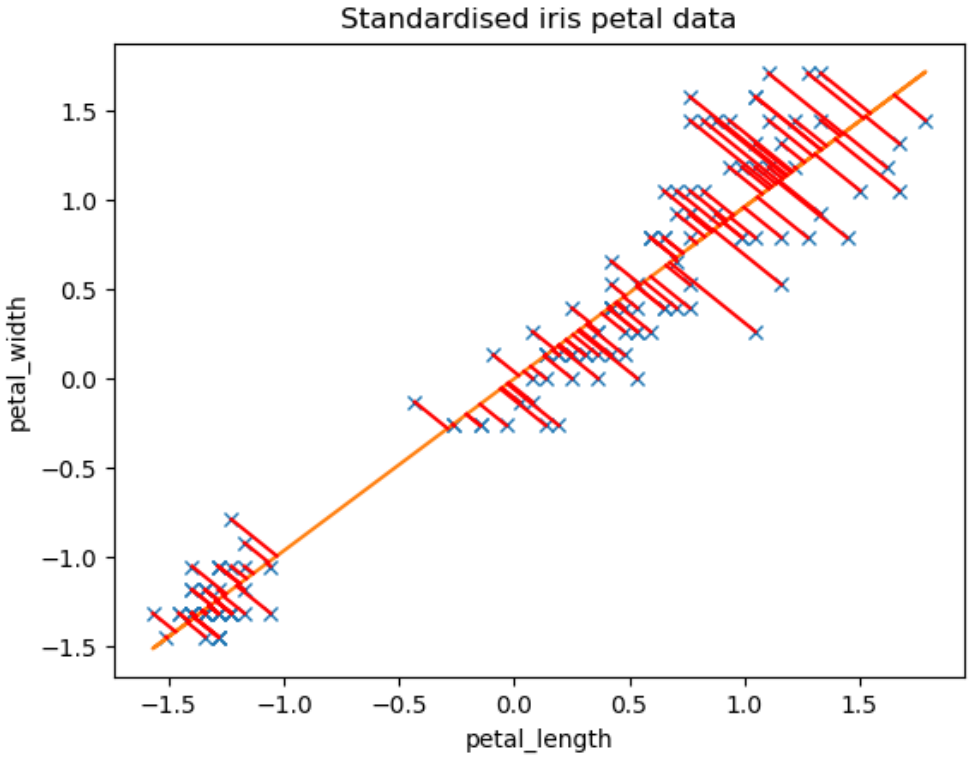


*Figure 8. Visualization of the principal component of standardized data of three different Iris flower species. The blue crosses are data points, orange line is the principal component, and the red lines point to where on the principal component the point closest to the respective data point is. (Fisher, 1936)*

As suggested by the graphical method above, principal components are variables constructed by mixtures of initial variables. The maximum amount of data is attempted to be inserted in the first component, followed by the second-most amount of data, and so on. From 10 variables 10 principal components can be created, though these principal components are not equal to the original variables in any way, rather they are an abstract representation of the data designed to conserve as much of the original data as possible.

To further illustrate the method, take 3 variables to apply PCA on, which can be plotted out in along x, y and z. Principal components are always 90 degrees to each other, exactly as an z-, y-, z-axis. What PCA does is to rotate the axis in such a way that one line is set in a position which maximizes the variance as described above. The principle remains the same for more than 3 variables. Note that before this can be done the data needs to be standardized and scaled. See figure 9 for an example of getting the second principal component on a model based on 2 variables.

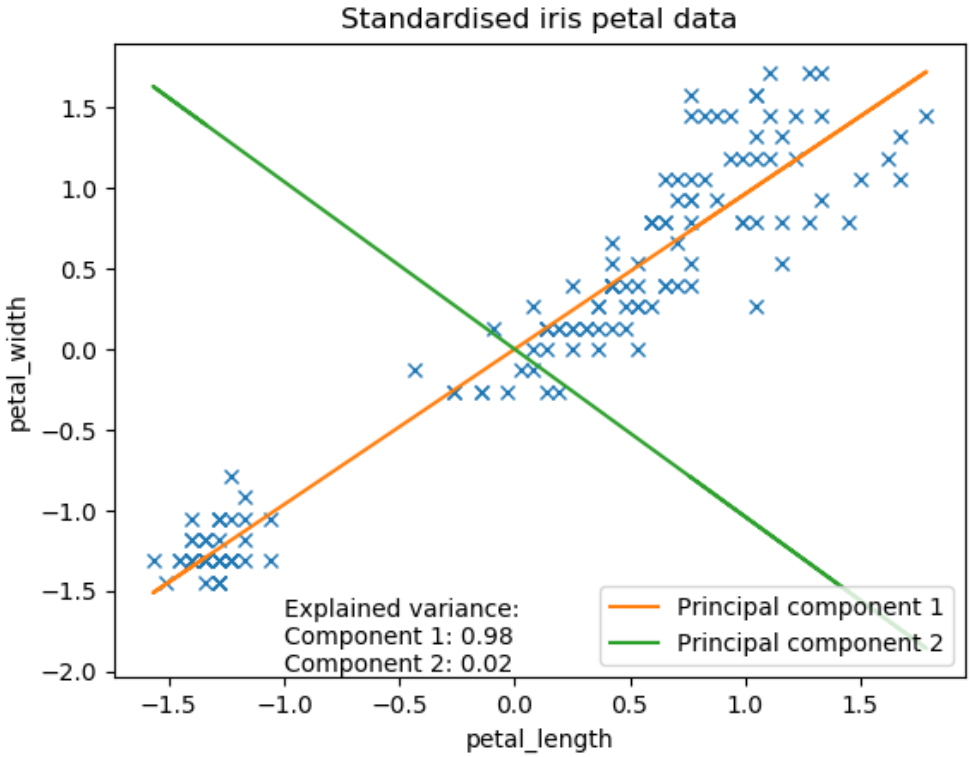


*Figure 9. Visualization of the principal component of standardized data of three different Iris flower species. The orange line is the first principal component, account for roughly 98% of the observed variance in the data set in the figure. The second principal component is the green line, perpendicular to the first principal component, and contains the rest of the information in the data set. (Fisher, 1936)*

The method was illustrated above graphically. The actual method used for calculating the principal components is done through standardizing the original data, calculating a covariance matrix from that standardized data, calculate eigenvector and eigenvalues from the covariance matrix, using that to determine the feature vector to finally transform the original (standardized) data to reorient it into a new axis. (Tharwat, 2016; Jaadi, 2019)

### 2.9.1 Standardization

PCA is highly sensitive to the size differences in ranges of the initial variables, e.g., a variable whose value ranges from 0 to 100 will dominate over a variable whose range is between zero and one. This will lead to biased results unless the variables are standardized.

Standardization in PCA is done through (22)

$$z = \frac{value - mean}{standard\ deviation} \qquad \textbf{(22)}$$

where z is the standardized value, value is the original value, mean is the mean of the variable and standard deviation is the standard deviation of the variable.

Standardizing means that all variables will be transformed to the same scale and become more comparable to each other. (Jaadi, 2019)

### 2.9.2 Covariance matrix

The covariance matrix is a method to see whether there is a relationship between different variables – if they are correlated. The covariance matrix is a symmetric matrix of size $p\,x\,p$, where $p$ is the number of variables. For each pair of variables, the covariance is calculated; if the covariance is positive the variables increase or decrease together, and if negative, one variable goes up as the other goes down. (Jaadi, 2019)

Covariance between two different variables can be calculated with (23)

$$\sigma(x,y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}) \qquad \textbf{(23)}$$

where $\sigma(x,y)$ is the covariance between variable x and y, n is the number of samples, and $\bar{x}$ and $\bar{y}$ is the mean for variable x and y, respectively.

### 2.9.3 Eigenvectors and eigenvalues

Using the covariance matrix, it is possible to calculate eigenvectors and eigenvalues, which in turn is used to determine the principal components of the data. Each eigenvector has an eigenvalue, and these can be calculated for each variable in the covariance matrix.

Eigenvector can be seen as the direction the variable has in each of the possible dimensions (where each variable makes up a dimension) whilst the eigenvalue can be seen as the length of this vector. The eigenvectors of the covariance matrix follow the line of the principal components, so calculating the eigenvector is the same as calculating the principal component. (Nicholson, 2020)

Once the principal components are calculated, it is possible to compute the percentage of variance accounted for by a given component by dividing the eigenvalue of that component by the sum of the eigenvalues of all components. By doing this, it is possible to rank the principal components in order of significance. (Jaadi, 2019)

### 2.9.4 Feature vector

By taking the ranked principal components from the previous step and removing the principal components with lesser significance (low eigenvalues), a feature vector is created from the remaining principal components. The feature vector is a matrix that has eigenvectors of the remaining components as columns. The information from the principal components removed will obviously be lost, but much of the original information remain. (Jaadi, 2019)

### 2.9.5 Creating new principal component axes

In the last step, the feature vector is used on the original data to reorient it to the ones represented by the principal components. This is done by multiplying the transpose of the original data set by the transpose of the feature vector. (Jaadi, 2019)

$$FinalDataSet = FeatureVector^T * StandardizedOriginalDataSet^T \quad \textbf{(24)}$$

### 2.9.6 Evaluating results of PCA models

A scree plot shows the amount of variance explained by a principal component in the PCA model in the order of the most to the least variance explained. Scree plots are useful when deciding how many principal components are necessary to use for analysis purposes and provides an overview of how well the PCA model summarizes the data. See figure 10 for an example Scree plot.

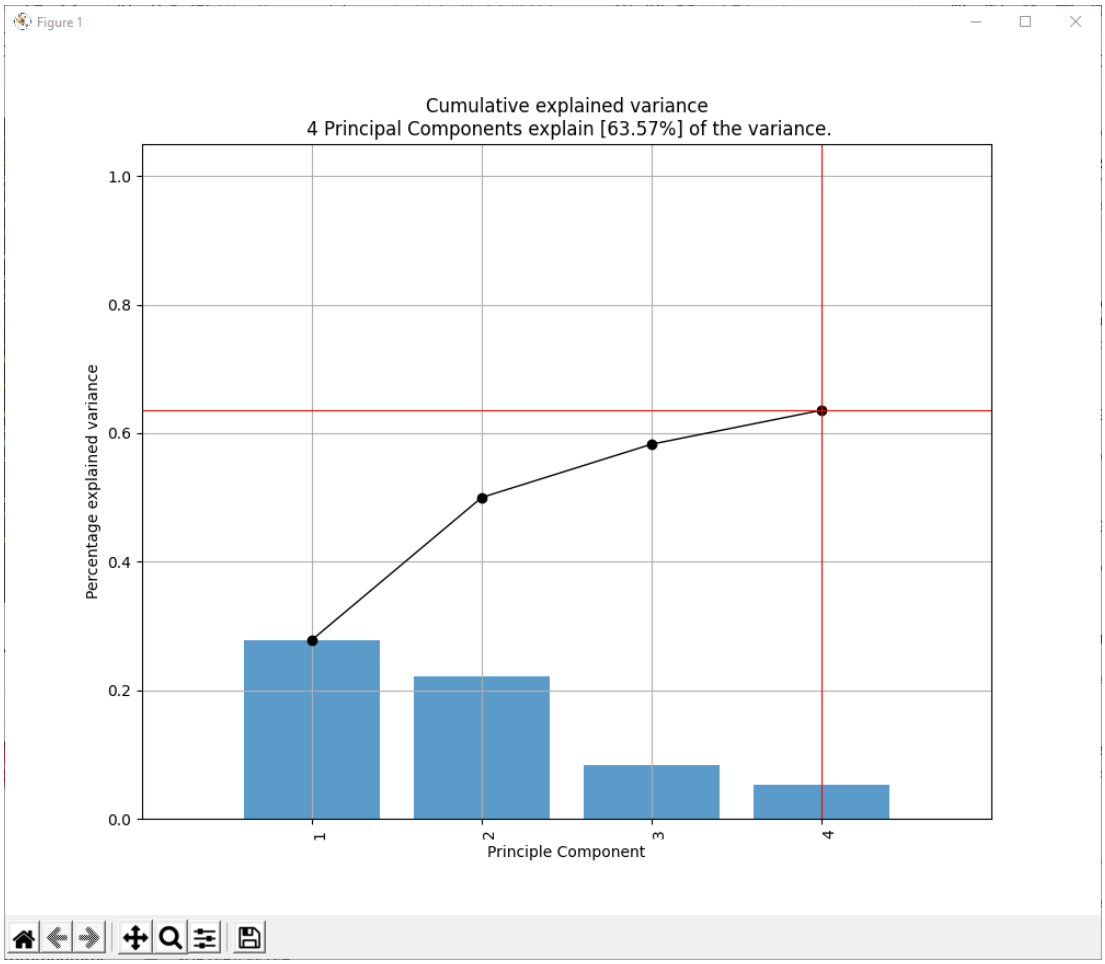


*Figure 10. Example Scree plot. For more info, see chapter 4.3.*

The score of a PCA model is the value that the principal components give to any given data point. This is the result of the PCA model, and plotting it out gives a score plot, from which it is possible to see how data points are scattered. Things to look out for could be whether any clusters are formed, or the data change according to some sort of category. For an example, see figure 11.

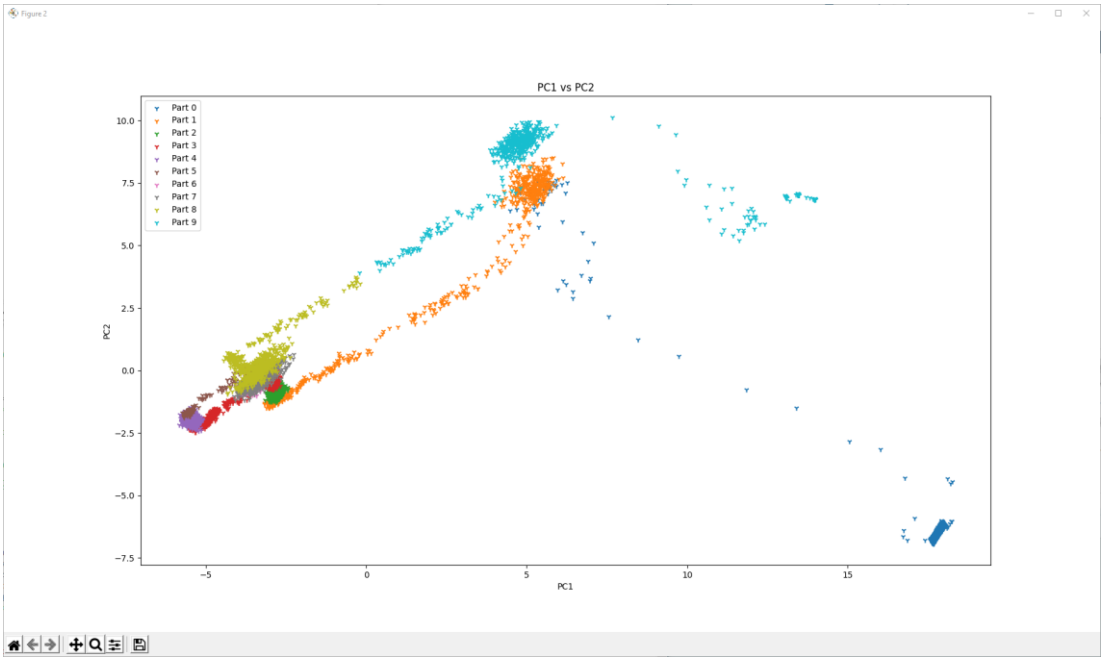


*Figure 11. Example score plot. For more info, see chapter 4.3.*

Loadings is a value of the direction of the data as compared to the direction of a given principal component. Loadings have a value between -1 to 1, where a 1 means that the data in the variable follow the direction of the principal component perfectly, and a -1 means that the data in the variable is in the complete opposite direction. This also means that variables with similar loading for a given principal component are correlated. It is of course possible to plot out the loading of two principal components; variables which are close together in such plots are of course likewise correlated. For an example, see figure 12.

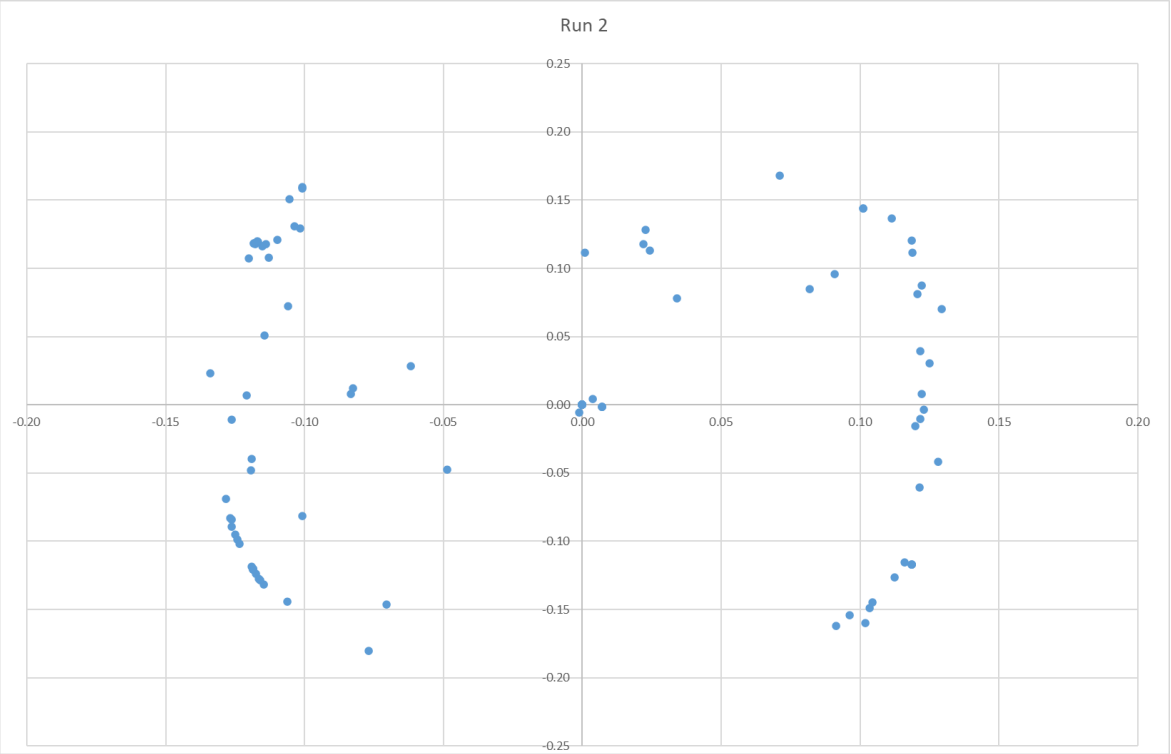


*Figure 12. Example loading plot. For more info, see chapter 4.3.*

(Wicklin, 2019)

### 2.9.7 The use of PCA

PCA offers some significant advantages. When plotting out score plots, it summarizes data in a way that is significantly more readable than using dozens or hundreds of variables at the cost of being somewhat arbitrary. Nevertheless, PCA can help human minds understand the big picture. PCA can also help machine learning algorithms in training if there are many variables involved in the training data, as it filters out "unnecessary" variables. The trade-off is of course that some data is lost, which might affect the theoretical upper limit of accuracy of predictions, though in cases where the number of variables is too much for a given machine learning algorithm, PCA could in practice increase accuracy.

Aside from reducing dimensionality (and thereby getting only the most important information from the data), some of the possible goals of PCA is finding relationship between observations, and outlier detection and removal. PCA is commonly used with unsupervised data, where it can help with data visualization, data reduction, and noise reduction in the data set. (Tharwat, 2016)

## 2.10 Machine learning

Machine learning is a method which has recently risen in popularity; recent advancements in both machine learning methods and processing power has made the technique viable, and an increase in the amount of data, especially of types not commonly collected before in a given application, which has led to a surge in demand for data analysis. Machine learning has the advantage of, in theory, letting computers make models for applications which range from trivial, to time-intensive, to next to impossible to create by other means.

The difference between a machine learning algorithm as opposed to a traditionally programmed solution is that whilst a normal program interacts with data to create an output, a machine learning model uses data and an initial model to get output, then attempts to change the model to get the output closer to a predetermined goal as defined by some measure. (Sarkar, et al., 2018) For an overview of machine learning techniques, see figure 13.

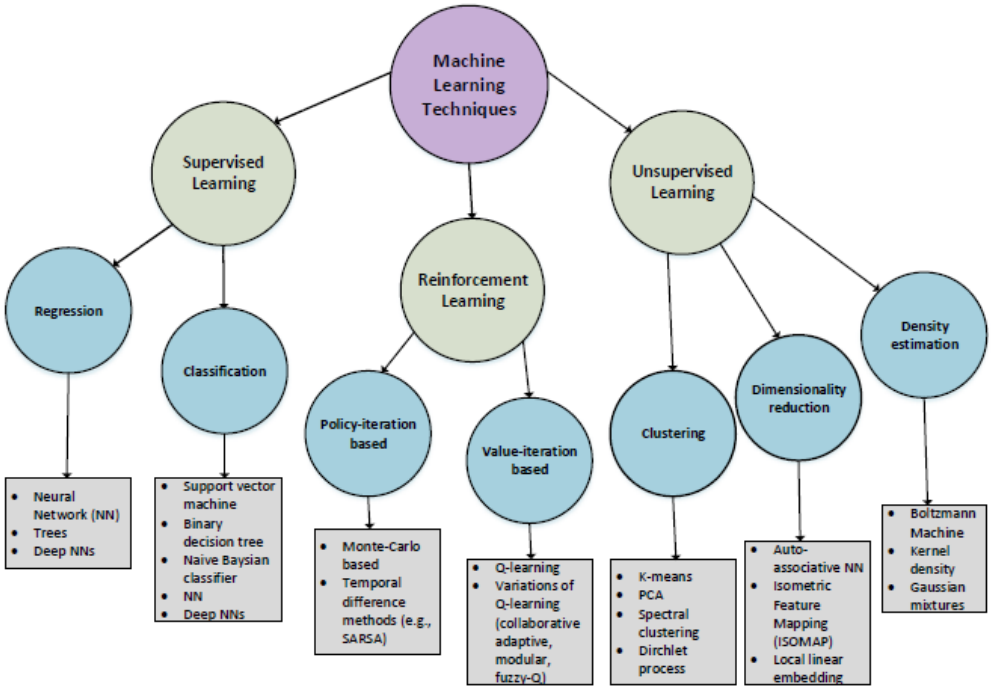


*Figure 13. An overview of machine-learning models and techniques. (Sharma & Wang, 2020)*

Given a set of data, it would be expected that there is an upper theoretical limit to the accuracy of a machine learning system, i.e., a model which is as correct as possible, but falling short due to patterns in the data utilized not being completely consistent. Depending on the machine learning method used, this theoretical limit can vary, but seeing as machine learning algorithms are still imperfect, it is likely that this theoretical limit will not be achieved in any non-trivial case. This means that attempting to help the machine learning things by for instance

adding additional metrics based on the input data could improve the performance of the machine learning algorithm.

### 2.10.1 Supervised versus unsupervised learning

Data labeled according to some criteria is often used in machine learning algorithms, either to make a classifier model, which has the goal of predicting different categories of data, or the labels themselves are used in the machine learning process. In supervised learning, the machine learning algorithm utilizes labeled data.

Labeling data can be very time-consuming if it requires human intervention, which means that using unlabeled data can be attractive. In unsupervised learning there are methods which can be used help the machine learning algorithm. For instance, a clustering algorithm can be used to identify groups of data points according to some criteria, then label the resulting clusters, which in turn can be used by a machine learning algorithm. Aside from clustering algorithms, clustering is also a common task to give to a machine learning algorithm. (Sarkar, et al., 2018; Google, 2020)

It is possible to have an entire machine learning model as a classifier, then have the output of that model as an input to another machine learning model. This way of using another machine learning algorithm to accomplish a secondary goal can in some cases help the primary machine learning algorithm to achieve its primary goal.

One of the methods used by the project team is neural networks, which was applied to a set of engine data.

### 2.10.2 Reinforcement learning

Reinforcement learning is different from both supervised and unsupervised learning in that it employs an actor which makes decisions, where the decisions it makes is designed to maximize a reward signal. Reinforcement learning does not use real data to power its learning (which is how many other machine learning methods function), but rather creates a policy (in a given state, make action x) which maximizes the measure as represented by the reward

signal. (Sutton & Barto, 2018) For more information regarding state and agents, see chapter 2.6.

Whilst a supervised machine learning model could have a task such as *given inputs x, predict outputs y* or *learn to classify these states from this labeled data*, and an unsupervised learning model would have a task such as *make clusters of different engine states* or *predict the probability density function of engine data*, a reinforcement learning model would have a task such as *maximize the fuel efficiency of the engine.*

### 2.10.3 Curse of dimensionality

The curse of dimensionality is a phrase which describes the growth of complexity of problems as new dimensions (or variables) are introduced. This also typically means an increase in computation time, for instance, if a calculation needs to be done between each pair of variables. Aside from the issue of computation, high dimensionality also creates huge issues when trying to cluster data, as the number of variables will act to make what will make up a cluster obtuse. (Sarkar, et al., 2018; Kuo & Sloan, no date)

The primary data used in this thesis has a total of 113 different variables. Some of the variables can be disregarded in some analysis cases due to their values practically never changing, but even with this simplification, a simplification which cannot always be done, a huge number of dimensions are present. Although not being a large enough number to make machine learning impossible, it might affect how good training results can get. Another issue is how complexity makes analysis done by humans significantly more difficult.

# 3 MATERIAL AND METHODS

## 3.1 Data analysis workflow

An example of how exploratory data analysis for creating algorithms can be done has been described in figure 14.

Exploratory data analysis is looking at the data in different ways and gaining an understanding of the data. This could mean visualizing the data through graphs, checking the data for inaccuracies or anomalies, and countless other data analysis methods dependent on what type of data it is. Exploratory analysis is about getting an understanding of the data, the quality of the data, finding faults or problems in it, and making hypothesis on the nature of that data. The project team used experience and intuition to preform exploratory analysis and discover findings.

If a problem with the data is discovered, the root cause is investigated. If this data was determined to not be a problem for the sake of data analysis, analysis continued as usual. If it was determined to be a problem, a decision on how to proceed needs to be made; either fix the root cause during the collection stage, fix it post-collection stage, or ignore the issue altogether, in that order of preference but limited to what is feasible considering the priorities of the project.

If a hypothesis of the data which might be useful to the project was discovered, the first step is to test said hypothesis. In case of the hypothesis being determined to be false, it is discarded, in case of the hypothesis being inconclusive, a new hypothesis is made and tested, and in case of it holding up through testing, the hypothesis moves on to further investigation.

The next step is a parallel process of defining requirements of the data for the algorithm to be valid and implementing these steps, as well as formulating criteria for decision-making. After criteria have been determined, statistical analysis is used to evaluate the criteria. After both processes are ready, the process moves on to implementing a viable proof-of-concept algorithm. After this is done, the algorithm is tested; if it is both works and is performant, the algorithm is ready to be implemented for testing in a real-life scenario.

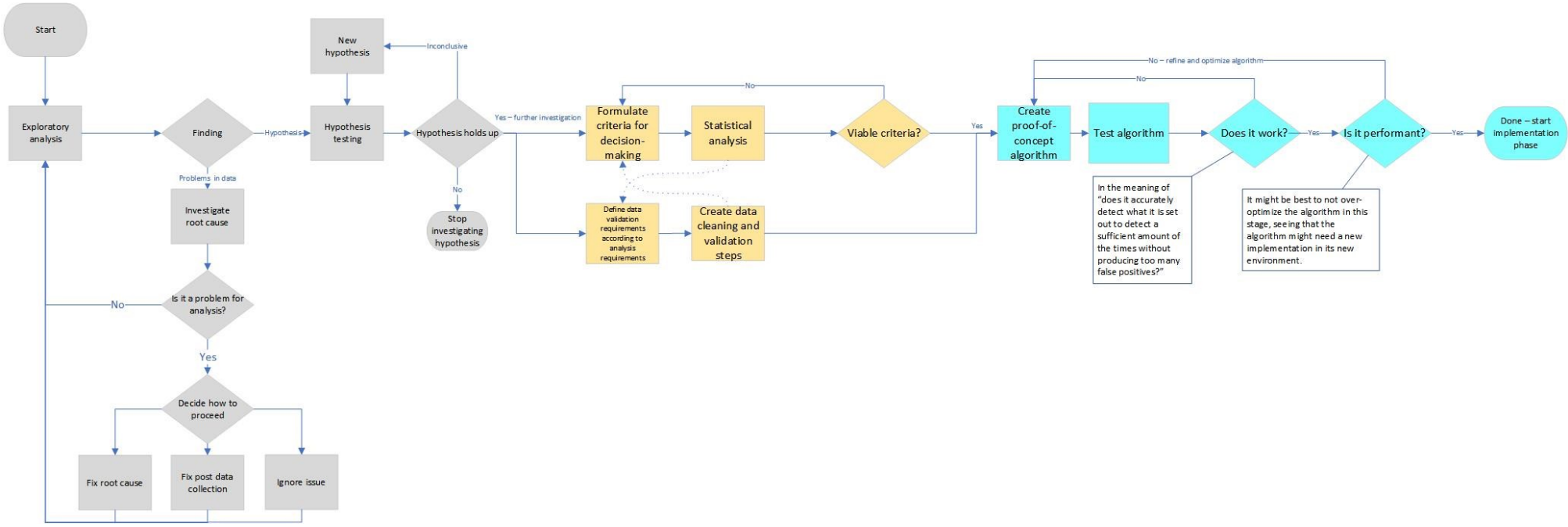


*Figure 14. A flowchart over the data analysis method used.*

### 3.1.1 Lack of data

The project team did not have much data of faults available throughout the project, and due to data ownership issues acquiring more data was not viable. This presented a major issue in that whilst data of normal operation was available, there was insufficient data of any given fault to make statistically significant assertions on how a fault of a given type could be detected.

## 3.2 System and equipment

The system studied in this thesis is an old laboratory engine. It has been modified numerous times in the past with incomplete records, which introduces additional uncertainty in working with the engine. There are 113 signals from sensors and the control system, with intermediate controllers and data acquisition devices leading to computers for data storage and analysis purposes. The signals in question are different performance signals (pressures, temperatures, amount of power etc.) as well as control signals. In addition, an edge device has been added to the system for the purpose of the project along with some new sensors that are external to the original engine system. The edge device was also given its own separate internet connection to facilitate data transfer to the cloud, where the data was shared to different teams working on the project.

## 3.3 Cases

Some of the work done during the project will be discussed in the following section. Background and methodology will be discussed in this chapter, whilst results and conclusions are discussed under the Results and Discussion sections, respectively.

## 3.4 Case: Investigation of timestamps in accelerometer data

As a part of the Edge project, a new model of accelerometer sensor was investigated. This model of accelerometer is not rated to match the usual requirements of accelerometers employed in the application used in the project. Furthermore, in an earlier version of the software used by the sensor, the sensors had issues with sampling rates, which made it

important to ensure that the sampling rate of the new version was sufficiently good.

This report describes the methods used in investigating the timestamps, as well as the conclusions drawn from said investigation.

### 3.4.1 Equipment

The data originated from two sensors of the same model. This model has several different sensor types integrated on the same chip, among which is an accelerometer. The model's intended use according to the manufacturer is for prototyping within IoT projects.

The accelerometers send their data through a router to a cloud storage, from where the data were retrieved. The sampling speed of the sensor during the sampling was set to 450 Hz, although other operations were taken by the sensor at slower intervals.

### 3.4.2 Data

For each measurement point, accelerometer data are stored in mG ($1/1000^{th}$ of gravity on earth) in the x, y, and z direction (not used in this report), as well as the timestamp in milliseconds since epoch. The data are stored in two different files, for each sensor respectively, the measurement points of which spans approximately 72 minutes and has 2159181 and 2159446 measurement points for the first and second accelerometer, respectively.

### 3.4.3 Methodology

The first step in the investigation of the timestamps was to calculate the time difference between each time stamp by taking the next timestamp in the series and subtracting the current timestamp from it (referred to from this point on as $\Delta t$) and visualizing these points.

Using this visualization, it was determined that most of the points vary between 1 and 3

milliseconds (as expected with a sampling rate of 450 Hz and with a resolution of 1 ms), though a number of points were determined to have a higher Δt than 3 milliseconds, one group at roughly 7-9 milliseconds, and a third one at significantly more than that. Another graph was created to investigate the time differences between these points and the creator of the software was consulted.

Finally, further investigations were conducted into the third group of measurement points, which will be referred to as extreme points.

At a later stage, further investigations on engine data were done. During this stage, the following formula was used to determine how much of the time sampling was delayed, to get a measure of the overall consistency of sampling rate.

$$\%\ of\ the\ time\ delayed = \frac{amount\ of\ time\ delayed\ in\ total}{duration\ from\ the\ first\ point\ to\ the\ last} \quad \textbf{(25)}$$

## 3.5 Case: Exploring and creating data treatment steps of data

One of the major sources of data in the project are from sensors inside engine, as well as accelerometer sensors from the same engine. During investigation of the engine, data faults were discovered which made analysis more difficult. Furthermore, after the data was inserted into a database it became relevant to the identify and pick out parts of the data in manageable chunks.

### 3.5.1 Requirements set on the data

The requirements set on the data ultimately comes from the requirements of the analysis methods used. The initial requirements are a clean set of data with timestamps that are verified to be good, though it is not out of the question that further data analysis will show need for additional data cleaning steps or improvements of these steps.

The project team use several different tools for data analysis, due to this fact, as well as limited access to the database, it was decided that the cleaned data should be put into CSV files, one

for each series of data.

### 3.5.2 Equipment

The work in this report was done using Python 3, primarily utilizing the module Pandas. Some visualizations not elaborated upon here were made with Excel.

### 3.5.3 Data

Two sets of data are to be cleaned in the data cleaning steps, first is the engine data, with a total of x variables noted down for each point, the second being the accelerometer data, with measurements in the x, y and z direction. Accelerometer data was not validated in this data pipeline, instead being validated in another step. The engine data in its original form has several faults which needs to be fixed before analysis.

### 3.5.4 Exploratory analysis

From previous exploration from the team, it was discovered that the sampling rate of the engine data is irregular, further investigation showing that any given variable seems to only change roughly every other row. Investigation of this phenomena was done by calculating the time difference of every point from the previous point, as well as seeing what variables changed had changed from the previous point.

Further investigation showed that the variables are divided into two different series with their own separate sampling rates independent of each other. The first series which contain a majority of the variables (henceforth referred to as series 1) has a sampling rate which gravitates towards 1219 milliseconds, whilst the second series (series 2) has a sampling rate around 1312 milliseconds (though with some variance for both series). For each row of data, only the variables of one of the series is updated; the variables in the other series of data are copied from the previous row as-is. This results in approximately half of the data in the dataset

being redundant. This also creates issues for time-based analysis, where if not properly accounted for, the data values would appear to only increase every other data point, when the data might be increasing steadily over time.

After having separated the data into the two series and calculated the time difference between points again it was also discovered that at some points the sampling rate would increase significantly within the series, down to under 20 milliseconds in between samples. The time difference between each occurrence seemed to be exponential, which might indicate some sort of glitch occurring in the system due to an integer overflow, but no further evidence was found to corroborate this hypothesis. These periods with high sampling rate seem to correspond to a single measurement point in the data, and even during these events none of the variables' individual sampling rate is faster than what is to be expected from the observed variance.

The work described above was done on small section of the full data in a text file format. The full data was available to be queried from a database, through which further experimentation and data exploration took place. Among other things, it was confirmed that the timestamps of the accelerometers and the engine data is in the same format, and the full extent of the data was explored.

### 3.5.5 Methodology

Different steps are required to clean the data from faults. Below, the data cleaning steps are described in the order in which they are employed.

### 3.5.6 Separation of data into engine runs

Seeing that it is unfeasible to put all data of the database into a single csv-file because of performance issues, it was necessary to, preferably as automatically as possible, separate the data in manageable chunks. It was decided that chunk would be a single running of the engine, from start to finish.

An algorithm to detect when, in the data, the engine was running was created; it would detect activity from the engine by looking at a single variable which a) is always zero when the engine was turned off and b) is always above zero when the engine was turned on. That way, to detect when the engine was running one only need to look at whether that single variable is above zero. In this case fuel consumption was chosen to detect engine activity. For each point where activity was detected, the engine was defined as running for +- 15 minutes to and from said point, and any active point within that 15-minute range further extend that range with their own +- 15 minutes range. This results in a time series which extend 15 minutes before to 15 minutes after the entire duration where an engine has been running (spanning over gaps of less than 15 minutes). This 15-minute buffer was chosen to give a baseline before the engine has started and after the engine has stopped if required in the analysis.

### 3.5.7 Separation of variables into series

Although the data strongly suggested there was two different series of variables that belonged together, it was not clear which variables belonged to what series. A second investigation was conducted to identify which variables belonged to what series.

First, for each variable independently, all points which had no change from the previous point was filtrated out after which the time difference from the previous point in the new filtrated array was calculated for each point. By taking the smallest time difference value found in all of these arrays it was possible to identify if its sampling range corresponded to the 1219 ms series or the 1312 ms series. For variables where the minimum time for a change was a multiple of the above-mentioned series, the series was decided by dividing with both of the series individually and assigning the variable to the series whose expected sampling rate matched more closely (i.e. divide the value by 1219 and 1312 for series 1 and 2 respectively, and assign the variable to the series where the result is as close to an integer as possible). Variables which never changed was assigned to series 1.

After the two different series were identified, the data was split up according to the two series.

### 3.5.8 Filtration of unrelated rows away from series

After separating the data into the two series, redundant data from the other series remained. At first a method which generated the expected series of the two sampling rates was employed to identify what rows correspond to what series, but due to the natural variation in sampling rate this method proved to be not viable. Instead, for all variables in a series and for every row, the total amount of values which has changed from the previous row was counted. If the total amount of values changed was more than 0, it was considered part of whatever series those variables were part of. If not, it was discarded.

### 3.5.9 Removal of time intervals with high sampling rate

After the filtration of unrelated rows time intervals with high sampling rate remained in the data. To eliminate these, consecutive points which were logged within a set number of milliseconds from the last one was identified and put in separate lists. After discussion with the project team, it was decided that only the last one of these points were to remain, seeing that the last point in each list contains all changes from the previous points. The rest of the points were discarded.

### 3.5.10 Reintroduction of data rows

In the filtration step it was assumed that all rows where no changes took place from the previous one did not belong to the series in question, but it is possible that for a full series none of the measured points involved changed. This can be seen as a gap in the data with a time difference which is a multiple of the expected sampling rate for the series. In the interests of having as clean a data set as possible, these points were reintroduced.

The approach to fix this was to reintroduce data from the original dataset wherever the time difference between points was more than twice as long as the expected sampling rate. For instance, is the time difference for a point in series one was 2440 ms, a single point should be reintroduced, and for a time difference of 7330 ms (approximately 6 times as long as expected),

5 points should be reintroduced. Each point to be reintroduced was searched for by, from the point following the gap, identifying the approximate average value where the missing point should be in the original data. Points were searched for in a range centered around this approximate average value. If more than one data row was found within this range, only one was reintroduced. If no data row was found within the range, no action was made.

## 3.6 Case: PCA analysis on engine signal data

PCA analysis is a method to summarize data by reducing the number of dimensions (or variables) it has whilst retaining as much data as possible by re-arranging the axes that make up the original data set. Seeing the fact that there are 113 signals logged from the engine installation, many of which are variations of each other and many more correlated to each other, PCA analysis could help in analyzing the data, both for humans and for machine learning algorithms. The largest disadvantage to PCA analysis, however, is that the new variables, called principal components, are arbitrary. In other word, although having a certain principal component at a certain value might mean something, this meaning has to be interpreted through analysis.

### 3.6.1 Equipment

The work in this report was done using Python 3, primarily using a premade PCA library (Taskesen, 2020) as well as matplotlib. Excel was also used for exploratory data analysis.

### 3.6.2 Methodology

The work initially started with compiling data from a set of runs into a single file. The end result was a file containing a set of 31 runs from 4.11.2019 to 7.7.2020. Data from each individual run was identified with a label containing the starting date and time of the run.

Analysis started out on the data set as a whole, where the scree plot, score plots, and loadings

for principal components 1-4 were analyzed. Afterwards scree plots were made for each run individually, and a few runs were picked out for more close analysis with score plots and loadings. Additionally, a shorter period within a run was analyzed.

Additional visualization of the results outside of what is included in the PCA library was achieved with custom code as well as the python library matplotlib.

# 4 RESULTS

## 4.1 Results: Investigation of timestamps in accelerometer data

The graph over all data of the run investigated for sensor 1 and 2 can be seen in figure 15.

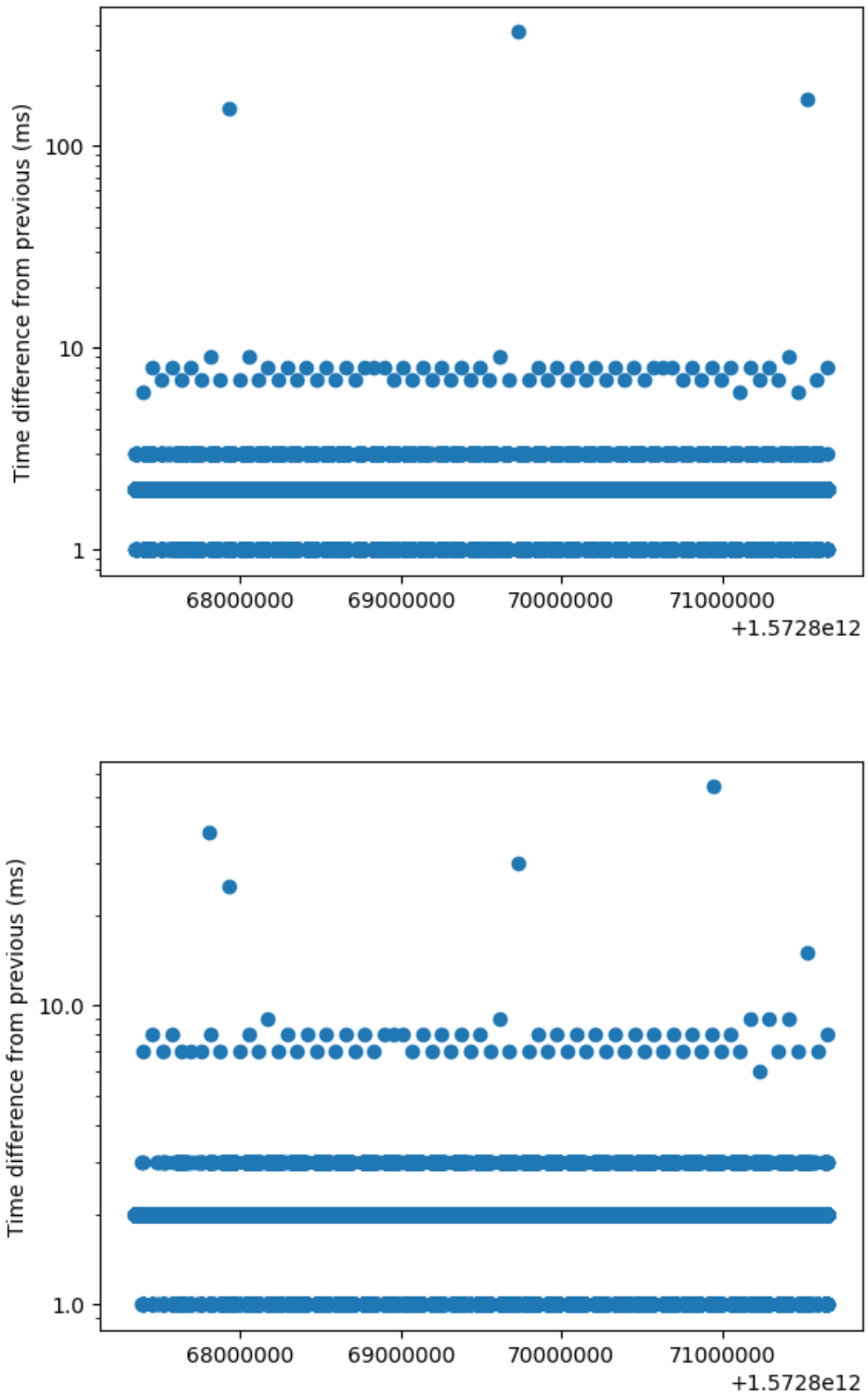


*Figure 15. The time difference from the previous point to the current point for each data point. X-axis is unix-time (ms).*

From these figures, the group of measurement points where the time difference from the last point is between 7 and 9 milliseconds appear to be periodical in nature. To confirm this fact, the points which were less than 4 milliseconds from the last point were filtrated out, and the time between each point was recalculated on the remaining data points. The result can be seen in figure 16.

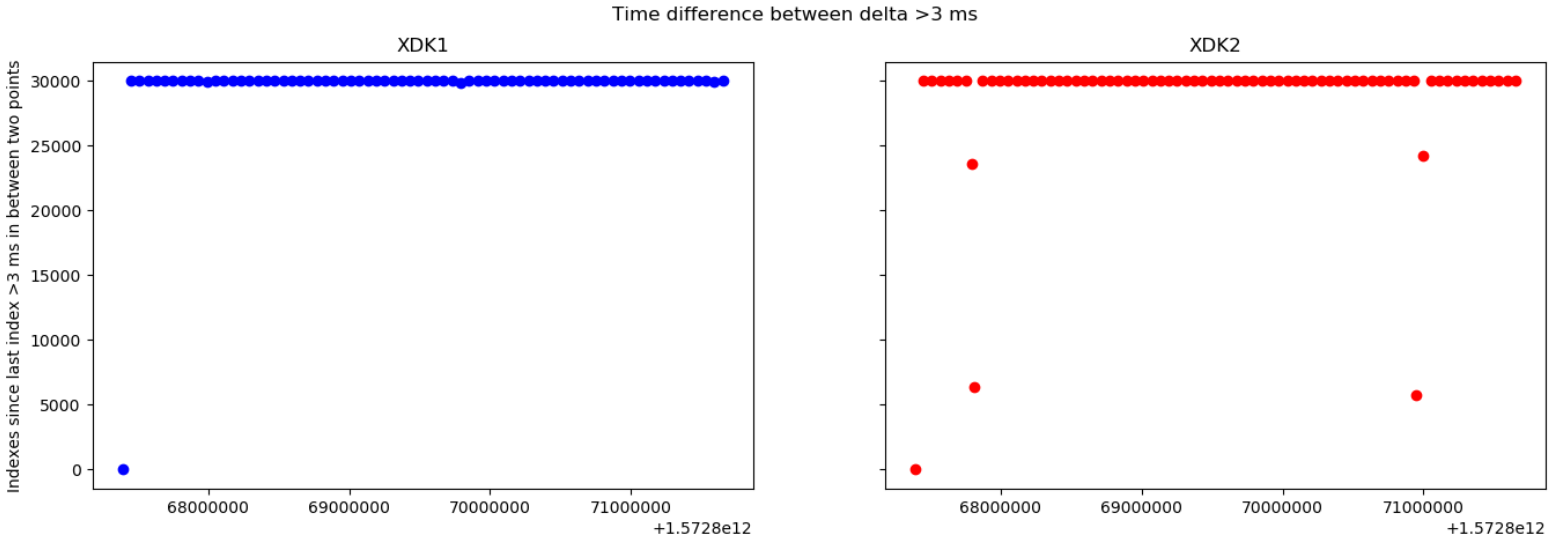


*Figure 16. Time difference between each point and the previous one, where all points with 10< ms to the previous point were filtrated out.*

As can be seen from the graphs, the points where the time difference is higher than 3 milliseconds, with very few exceptions, happen every 30 000 indexes, or every 60 000 milliseconds – every minute. This includes the extreme points, except for 4 points on the second sensor, which are caused by 2 irregular points in between two regular points. By further eliminating every extreme point (see figure 17) and redoing the calculation for Δt, Δt for all points in that

series is set at either 1 minute or 2 minutes. In other words, this means that aside from the two irregular extreme points, every point which has a delay larger than 3 milliseconds happens a minute apart from each other.

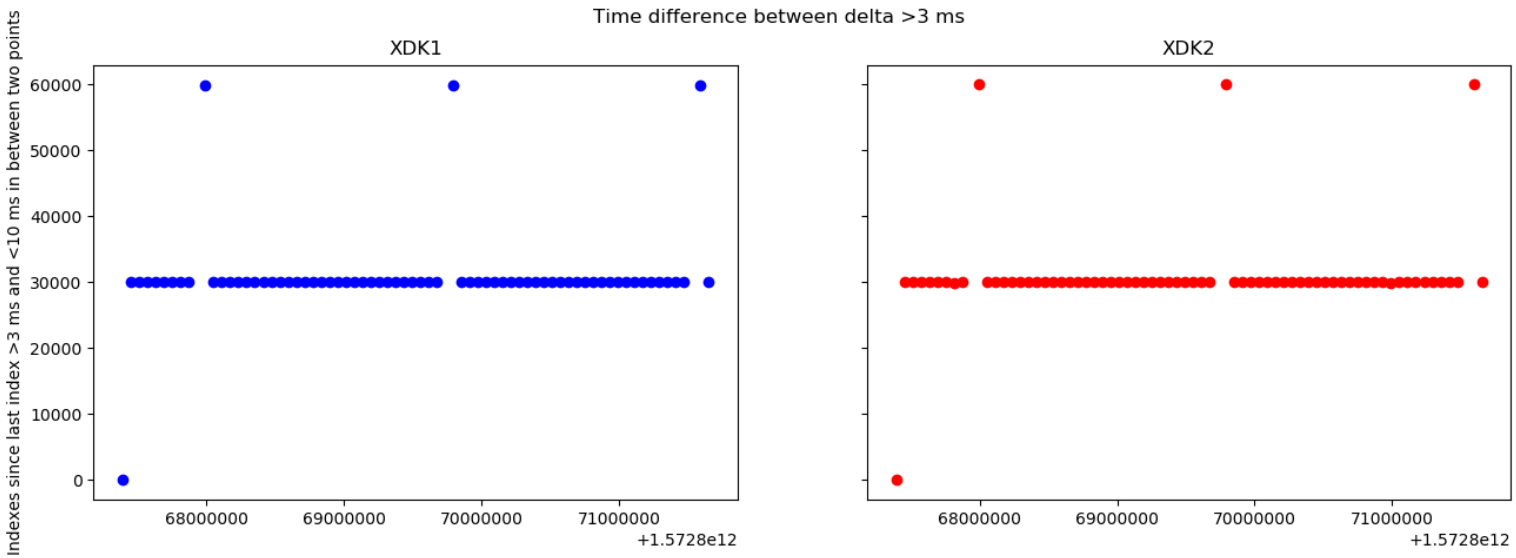


*Figure 17. Same as figure 16 except with irregular points filtrated out.*

After this, a closer investigation of the extreme points was made., see table 1 and 2.

*Table 1. Extreme points on sensor 1.*

| Delta t | Timestamp |
|---|---|
| 154 | 11:45:31.862 |
| 367 | 12:15:31.862 |
| 170 | 12:45:31.862 |

*Table 2. Extreme points on sensor 2.*

| Delta t | Timestamp |
|---|---|
| 38 | 11:43:19.888 |
| 25 | 11:45:32.708 |
| 30 | 12:15:32.708 |
| 54 | 12:35:44.130 |
| 15 | 12:45:32.708 |

As can be seen from the tables, the points where which take longer than 10 milliseconds happen with extreme regularity, every half an hour down to the millisecond (not including the two irregular points). By taking the start points in the table above and adding the $\Delta t$, it is possible to calculate the end point of all these points and conclude that although the extreme points are close to each other in time, they do not overlap. This time difference could be due to how the software handles things, and not an actual error.

The developer of the sensor software was consulted at this point. Asking about the patterns in the data, the software developer confirmed that every minute, additional measurements were taken, and every other minute one more measurement point was taken, which matches the increased time between points almost perfectly. The regular extreme points were concluded to, most likely, coincide with clock synchronization with an NTP server, although no explanation was found for the irregular points. The author of the software did not make any guarantees that timestamp data could be synchronized between the two sensors. Also note that sensor 1 takes a lot longer to execute clock synchronization (the extreme points) than the second sensor.

The analysis process was repeated on other sets of data, after which flaws in the data became apparent. For an example of the result of a later run see figure 18.

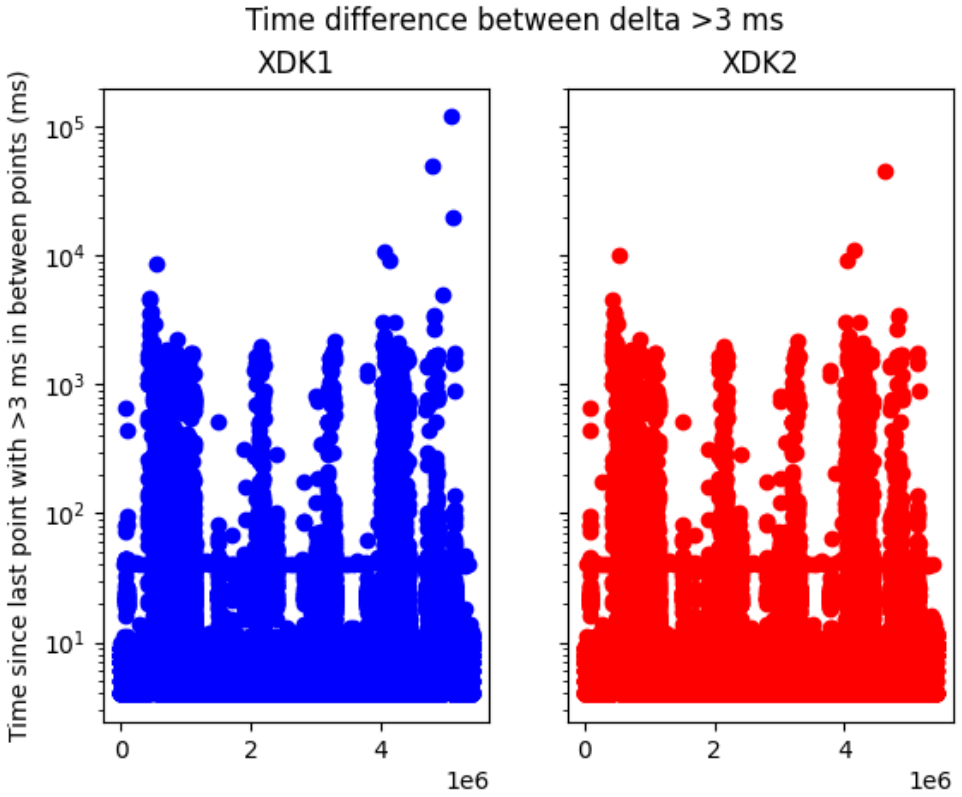


*Figure 18. Time difference of points for an engine run. Note the band at 40 ms for both sensors, as well as similarity in shape. X-axis is milliseconds since start of the time series.*

To measure the severity of the delays, the percentage of time the sensors were delayed was used as a measure according to (25).

Investigating the time during a run as opposed to the period 15 minutes before and after that period shows a statistically significant difference, see table 3. This could imply that something systematic happening before, after, or during the run affects the measurement system somehow.

*Table 3. The amount of time all runs were delayed in % during the run, and 15 minutes before and after the run.*

| Measure | Before run | During run | After run |
|---|---|---|---|
| Average | 22 % | 14 % | 28 % |
| Median | 3 % | 9 % | 20 % |
| Uncertainty of average | 4 % | 2 % | 3 % |

Plotting the amount of time each run was delayed results in figure 19.

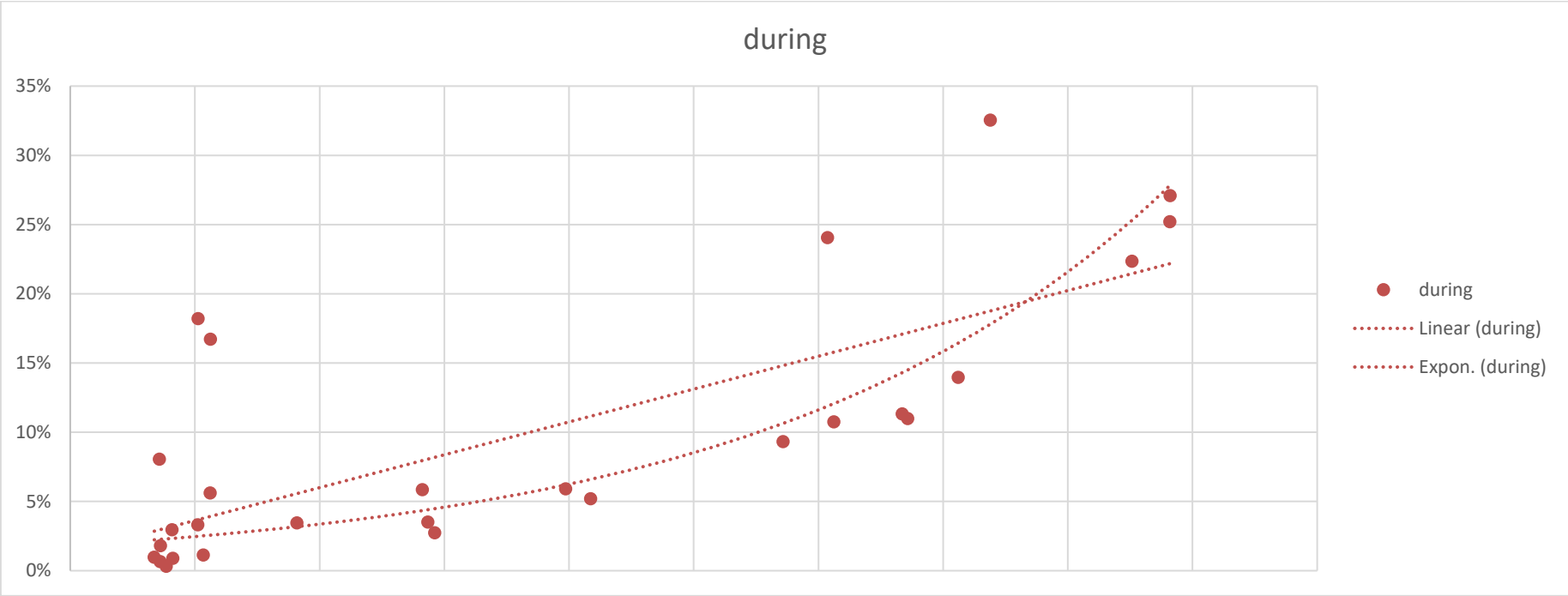


*Figure 19. The % of time each run was delayed during the run, x-axis is the date the run happened. Note that some early runs had technical issues which were later fixed. Linear and exponential curve fits for visual aid.*

There is an obvious trend of degrading performance over time, despite the promising initial results.

## 4.2 Results: Exploring and creating data treatment steps of data

Using all data points available of fuel consumption, a list of active points was created, and fed to the algorithm which identifies start and stop times of engine runs. The algorithm successfully found a set of different start and stop times corresponding to an engine run. These start and stop times were cross-referenced with a log of engine runs made and was used in further steps to pick out data from specific engine runs, both for the accelerometer data and the engine data.

At first, the above method for separating things into series was utilized in the hopes of creating an automated data pipeline which could consider changes in sampling rates and data variables. However, this proved unreliable due variables that changes slowly or not at all being inconsistently classified to one series or the other. Instead, the above method was used to identify a list of two variables, which would always remain the same. Later, a check with a partner within the project partially confirmed the list by looking at engine installation documentation, and a revised list was created and used.

Filtrating rows and removal of time intervals with high sampling rate produces mostly clean data, except for gaps in the data. After reintroduction of missing data, the data overall seems to be clean, with only a few (<5) points in each series having a time difference outside the natural variation of the time series for each engine run (where each engine run has thousands of points in total).

The finished data was then saved as 3 different CSV-files, one for series 1, one for series 2 and one for the accelerometer data (where the only change from the original data set was a column signifying from which sensor the data originated).

## 4.3 Results: PCA analysis on engine signal data

Overall, over most runs, the principal components 1-4 accounted for most of the variance in the data. Individual runs had an average of 83.42% of data explained in principal component

1-4, and when using PCA analysis on all data at once the same percentage was 63.57%, see figure 20.

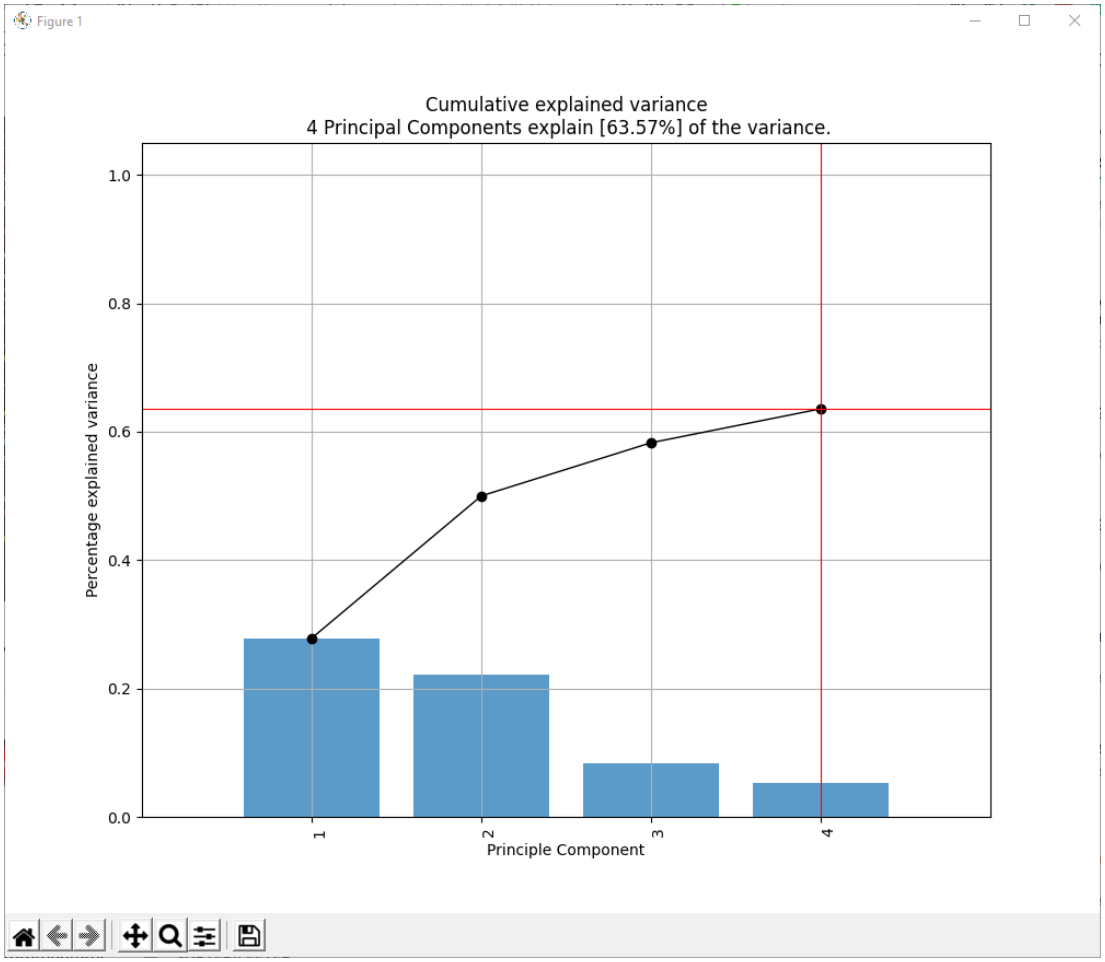


*Figure 20. Scree plot of model created using data from all runs.*

To visualize how data moved over time in the score plot, data from individual runs were split into 10 different chronological parts, see figures 21 and 22 for the first two principal components of two runs. After comparison with engine load, in some cases, it is apparent that principal component 1 is highly correlated to engine load, appearing to move up and down with engine load, then stabilize over time (see figure 23). This suggests that change in engine load accounts for most of the variance in the data, at least when the engine is not running stably. There is still major variation in the shape of data in score plots, which makes sense considering how sensitive PCA is to input data. This also means that some sort of state can be determined from PCA analysis, i.e., roughly what engine load is currently used. Any further

deductions of what can be determined about the state of the engine from the score plot needs further analysis.

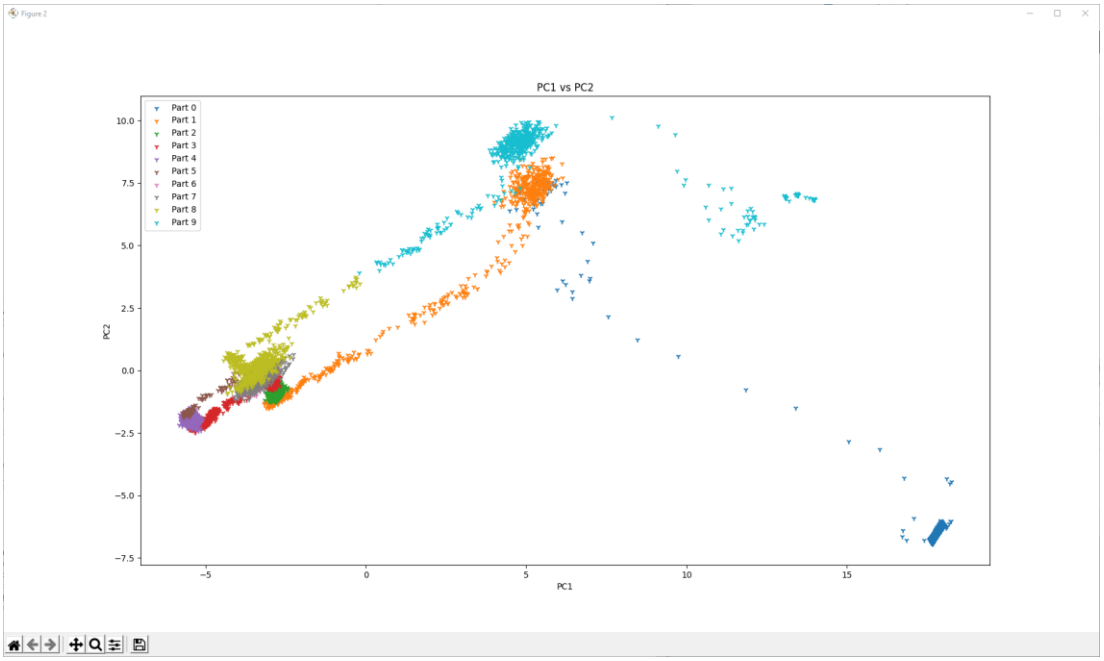


*Figure 21. The score plot of two different principal components of a run, split into 10 chronological parts.*

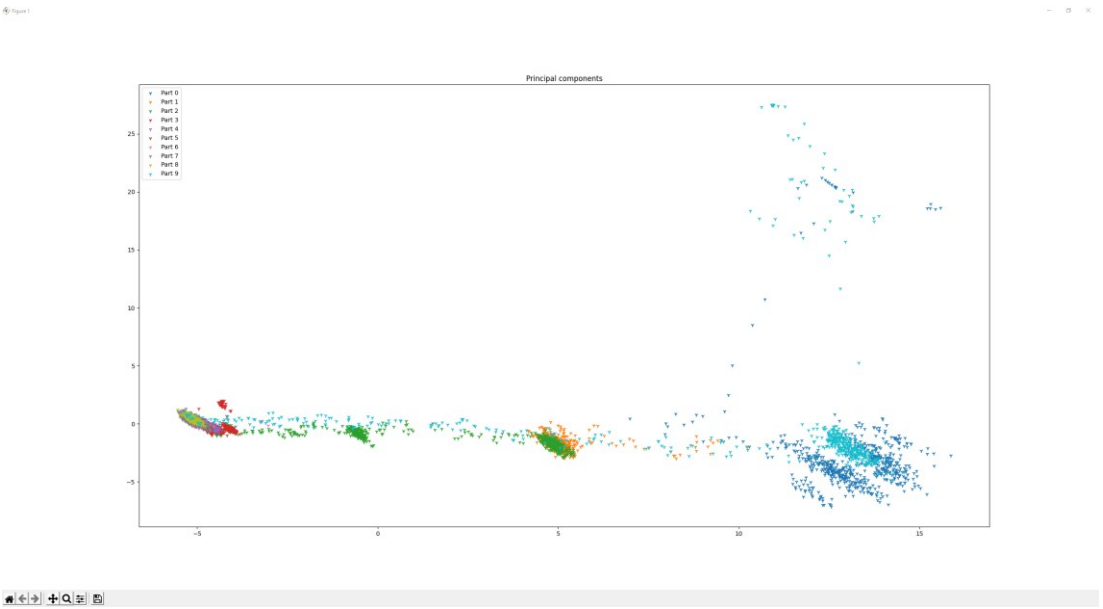


*Figure 22. Another score plot of two different principal components of a run, split into 10 chronological parts.*

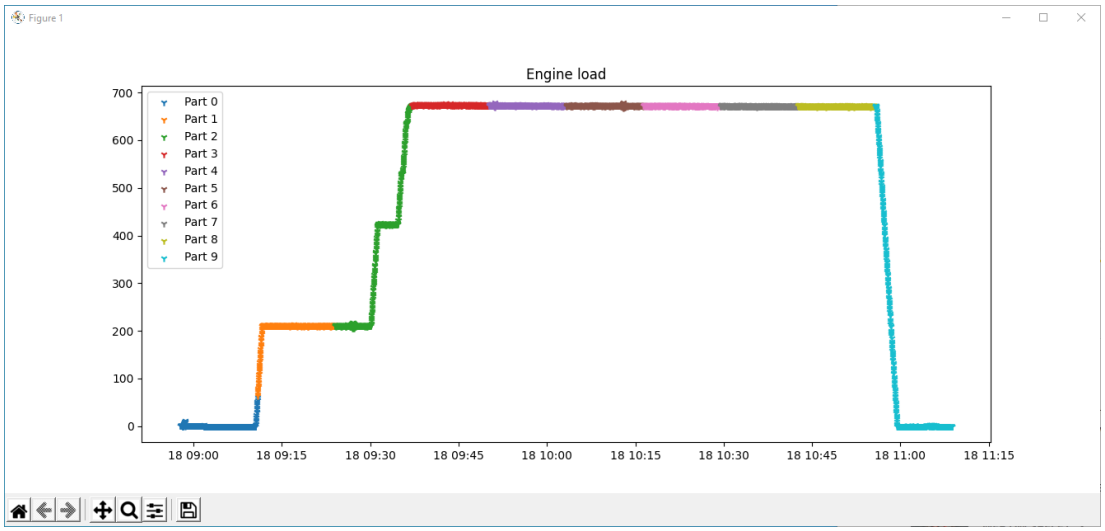


*Figure 23. Engine load split into 10 chronological parts from the same run as in figure 21. Note how in this model, the plot moves to the left as more load is added and appears to stabilize after having reached a step change.*

Individual runs overall have a lot more variation explained in the first 4 principal components than a model trained on all data. This implies that there is significantly less variance in different variables inside of runs than there are between runs. This makes sense seeing that the data in question was the result of test runs with a multitude of different test parameters, as well as natural variation between test runs in at least environmental factors, but also likely the state of the engine overall.

During the test run, an apparent anomaly was detected, see figure 22. The source of the anomaly was discovered by looking through the loading and determined to be caused by a set of 4 related signals, see figure 24, 25, and 26. One of the signals corresponded to PC4, and the rest of the signals to PC3.

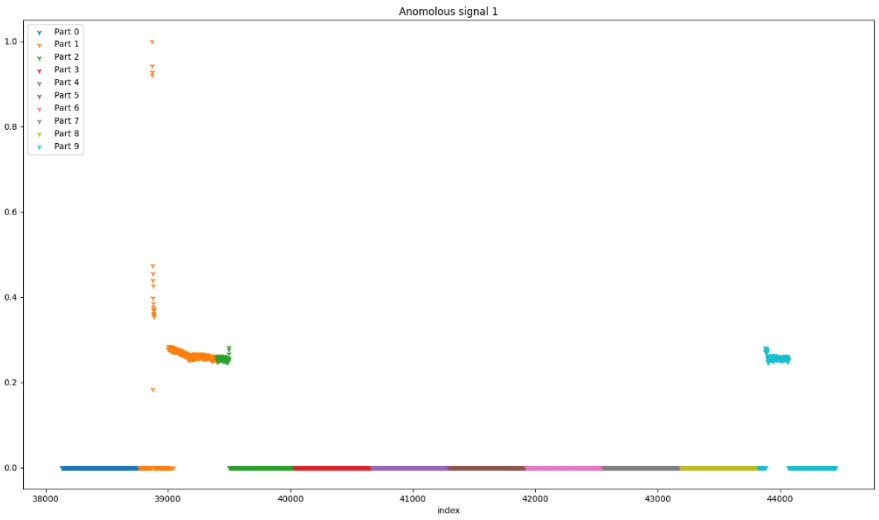


*Figure 24. An apparent anomaly in PC3 and 4 from an individual run.*

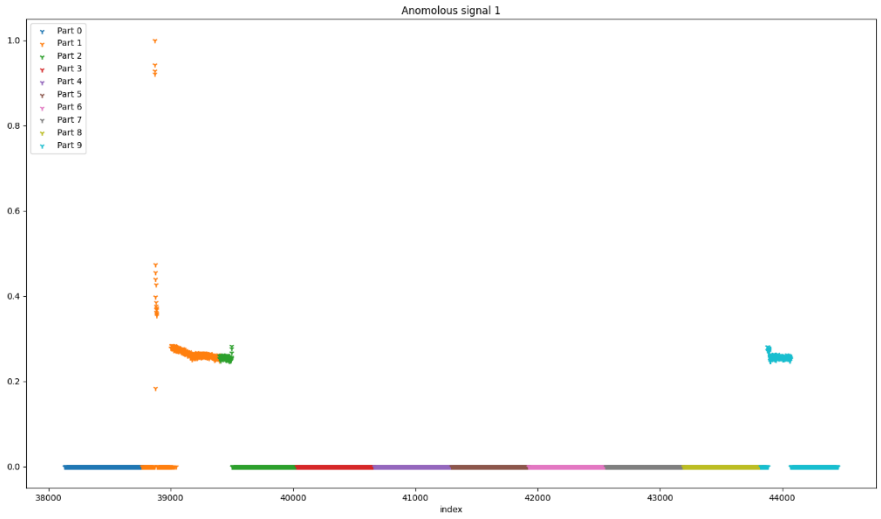


*Figure 25. Anomalous signal which corresponded to PC3.*

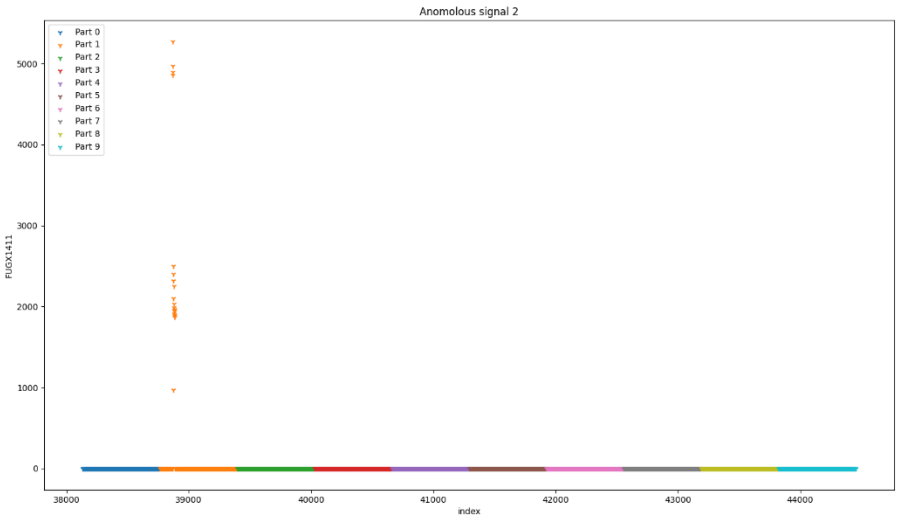


*Figure 26. Anomalous signal which corresponded to PC4.There was another 2 signals with virtually the same spike.*

When running on all data, the resulting score plot can be found in figure 27.

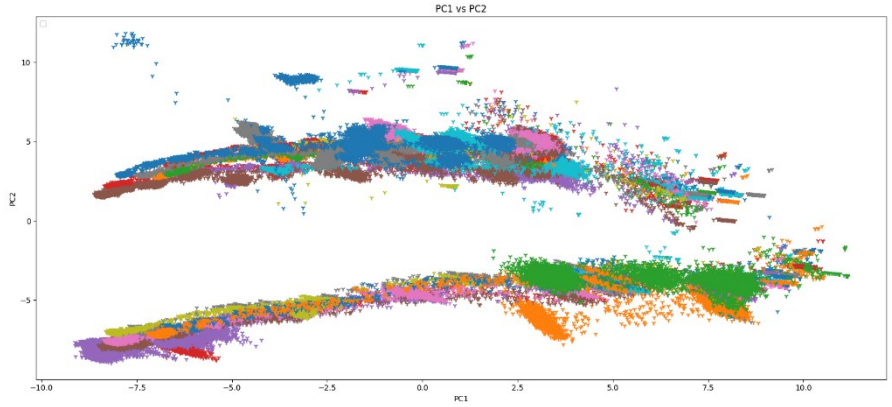


*Figure 27. Principal components 1 and 2 for a PCA model trained on all data. Each color represents a different run, and some colors are used more than once.*

In the data, there are two different clusters divided by principal component 2. PC2 > 0 was taken as a rough estimate of the point where a shift happened from one state to the next, and a table of the number of points PC2 > 0 can be found in table 4. The shift happens around run 13 and 14, after which all points are at PC2 > 0.

*Table 4. Table of the number of points >0 for PC 2. Note that runs have a differing number of data points.*

| Run number | Amount > 0 for PC2 |
|---|---|
| Run 1-11 | 0 |
| Run 12 | 5 |
| Run 13 | 196 |
| Run 14 | 5463 |
| Run 15 | 6460 |
| Run 16 | 11302 |
| Run 17 | 5699 |
| Run 18 | 8416 |
| Run 19 | 5650 |
| Run 20 | 5562 |
| Run 21 | 2362 |
| Run 22 | 3740 |
| Run 23 | 3937 |
| Run 24 | 3937 |
| Run 25 | 1033 |
| Run 26 | 7831 |
| Run 27 | 3768 |
| Run 28 | 4948 |
| Run 29 | 856 |
| Run 30 | 1962 |
| Run 31 | 11803 |

By cross-referencing data, it was determined that the shift from where PC2 < 0 to PC2 > 0 corresponded with a routine maintenance task. This was another case where a state change was clearly visible in the PCA model.

As mentioned earlier, engine load seems to account for most variability in all the runs in the data set. What phenomena or properties of the engine the next principal components correspond to are not as obvious and is likely to be shifting from data set to data set depending on how the engine is ran. In fact, this is obviously the case in the run with the anomaly, where a single rare event caused a set of variables to become more prominent than usual.

Loading for several runs were analysed. In loading plots, points close to each other are correlated. Several series of variables were found to be correlated to each other, where load and reference load were found to be highly correlated, and other measures, like pressure, also correlated to load to a lesser degree. Signals which were variations of each other were also found to be correlated to each other. Loading plots were compared to engine loads to achieve the purpose of a biplot, i.e., seeing what variables corresponded to pulling the data in what direction for a given data set.

The data from an individual run was taken from a point where the engine just reached a new load level, until just before that load level was changed again. The loading plot of this was compared to the load level of the entirety of that run, see figure 28 and 29. Surprisingly, it appears that the shorter time frame, despite being constrained to a single load level, where the full run contained significant load changes, contained more variance. As expected, the relative grouping variables were different as compared to the loading plot using all data from the run, which suggests that the data from the full run does not have the same distribution as the section of the same run.

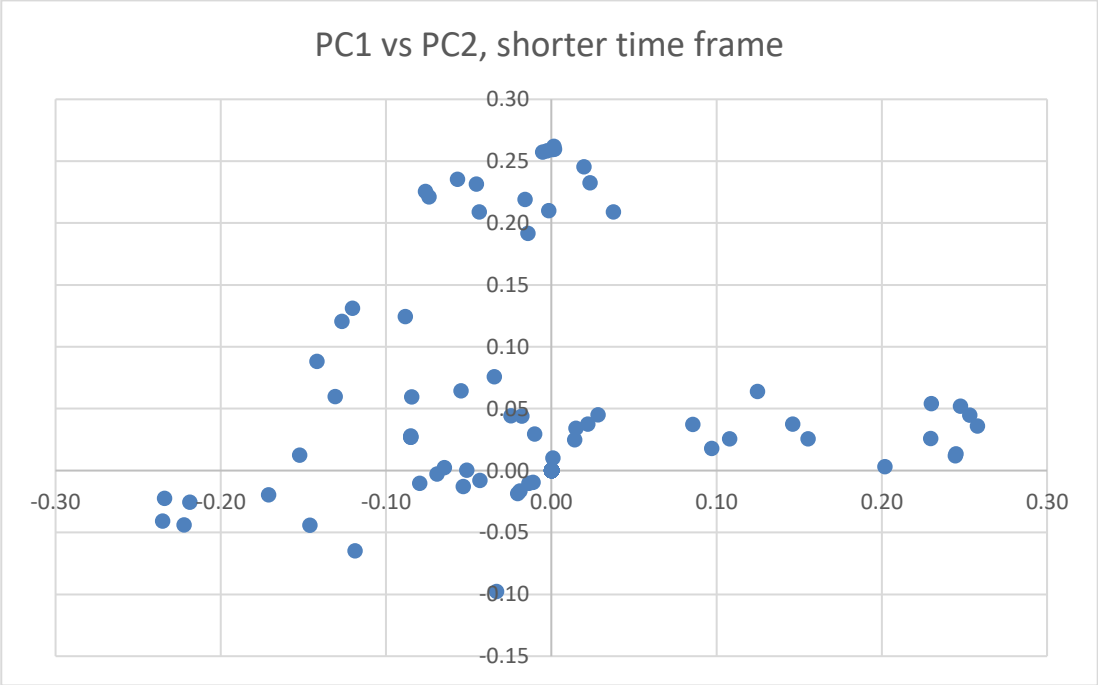


*Figure 28. Loading plot of a principal component model made from a section of a run.*

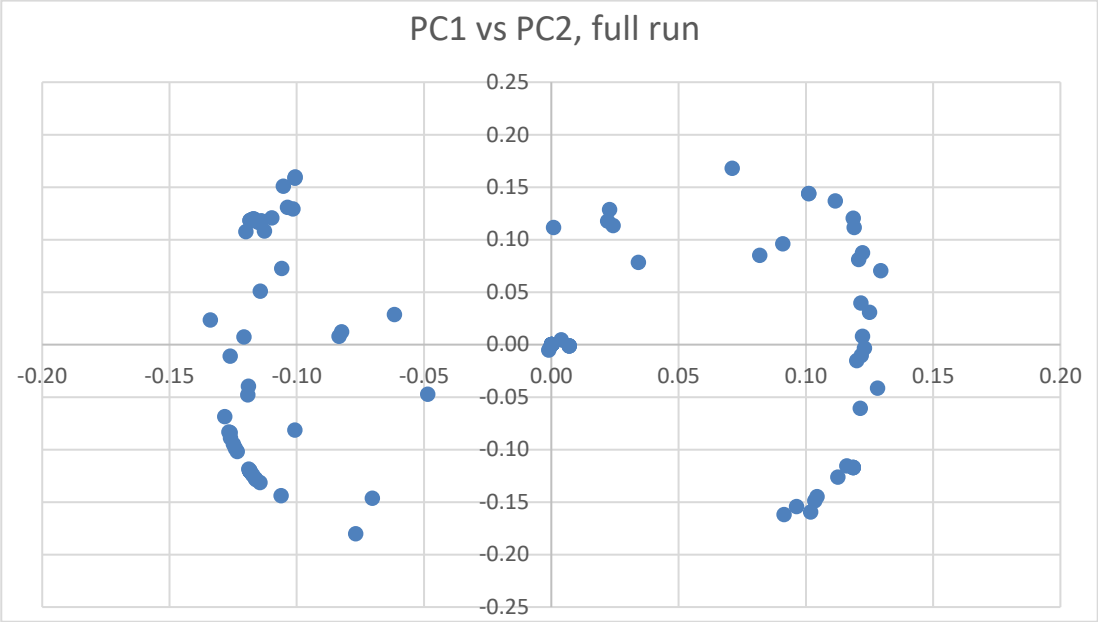


*Figure 29. Loading plot of a principal component model made from an entire run.*

### 4.4 A model for categorizing state

One of the largest challenges in analyzing the data set in the project is that what constitutes a state of normal operation and what constitutes a fault state is unknown. The only solution to this is to get more data, and do more analysis, which was both done throughout the project. However, due to the complexity in the data caused by the curse of dimensionality, evaluating every possible combination of variable values as a unique state is next to useless. A possible solution to this issue will be described in this chapter.

#### 4.4.1 Assumptions by the model

This model has a few basic assumptions. Firstly, it is assumed that the system is deterministic, which in this context means that given that the value of each variable affecting the system except one is known, the last variable can be determined. All the variables which affect the system make up the real state, or the real world in all its complexity; enough information that if the situation were precisely recreated, and assuming the system was deterministic, it would

behave the same as the original system.

The second assumption is that there are known variables that are a part of the real state, for which at least approximated values of are available, which affect the system but are not affected by the system. Given that the system is deterministic as described above, these variables can be used as input values in a model.

The third assumption is that the input values can also take the dynamics of the system into consideration. To clarify, assume that the system has a static part, including but not limited to the overall wear and tear of the system, which is mostly unaffected, at least in the short term, by dynamic parts. For instance, heating an engine up to operating temperature then cooling it down to its previous temperature would not affect the static parts of the system (assuming it is done in a way which does not harm the engine). Metrics based on the input variables can be added to properly take dynamics into consideration.

The system would have measurements of these input values, as well as measurements of some of the variables which make up the real state of the system. The rest of the variables in the real state are unknown. Given the above, it is possible to reason about the real state by seeing how it is affected by the input variables. For a given real state, giving the model the same input variables should generate the same result each time, where a part of that result is measured. For a given set of input values the rest of the known (measured) values in the real state can be noted. If, for the same input values, a distinctly different result is observed in the known state, the real state of the system must have changed.

Seeing that what is measured is often dynamic values, and what is interesting in terms of determining the state of a system is often static values, it would be possible to at least in some cases conflate changing known variables for a given set of input values with a change in the static state. Conversely, observations with similar known variables for a given set on input values could be in the same state.

This observable state is not the same as a Markov state, the latter of which are often categorical in nature, such as normal mode a, normal mode b, fault mode a, fault mode b etc. Rather, these states can be mapped to different Markov states (which are defined in a manner which is suitable for the task which is to be achieved).

### 4.4.2 Application of the model to an engine installation

Applying this model to an engine installation, the first thing is determining the input variables. Since an engine installation is a controlled system, using controlled and environmental variables as inputs for the model should be reasonable as they affect the system, but are not affected by the rest of the system.

Controlled variables are variables directly controlled by an operator. This would most commonly be load but would vary depending on the system in question. Environmental variables are variables such as outside temperature, which would be due to the weather, rather than the system itself. Only environmental variables which are not affected by the system should be included as an input variable.

By using the input values and matching up their corresponding output values it is possible to take a guess about the state the engine is in by looking at the probability of that state outputting that same value, as in a hidden Markov model. Assuming that the information is sufficient to determine state with any degree of accuracy, and that enough information of the system is provided to achieve the Markov property, the probability distribution of the next state should be able to be determined.

The static state is assumed to remain virtually the same in the short term. This assumption helps model creation in that data points close to each other in time can be assumed to have a similar static state, which can then be grouped together. These static states which are grouped together can be investigated to determine what sort of outputs are expected from a given static state, what sort of variance it has, etc.

Another use of categorizing data with the input variables is the fact that it can be used to determine whether the system has been in a given situation before. For instance, outside temperature could be at an all-time high, or operators might decide to drive at a higher load than ever before. In either case it would be trivial to determine that this situation is new, as opposed to a naiver system which would sound an alarm of a possible change in the state of the system when both examples above do not indicate anything about the state of the system itself.

# 5 DISCUSSION

## 5.1 Accelerometer timestamp data conclusions and further actions

The nature of the timestamps of the two sensor systems were investigated. In the initial results, the causes of slower sampling times were all explained, aside from two irregular points.

The longer $\Delta t$ of points where more measurements of different types were taken, suggests that although the sensor can handle the normal workload just fine, any additional measurements cause delays in the usual accelerometer measurements. In the long term, this can limit the use-cases for the sensor model.

A few explanations were proposed to sensor 1 appearing slower to synchronize time than sensor two. One explanation is that the software might have slightly different versions from each other, and the other one being that there are hardware differences; maybe the clock of sensor 1 is significantly slower than the one in sensor 2 and will therefore make a larger jump as the clock is synchronized.

Due to prioritization in the project, it was decided that the causes for the irregular extreme points and the fact that synchronization appears to be slower in sensor 1 should not be investigated further. Initially, the timestamps of the data were decided to be good enough to meet the requirements of our analysis, seeing that any potential errors caused by the observed faults in the data were small.

The second analysis showed a completely different picture, with significant degradation in data sampling rate. A significant amount of data was proven to be unusable due to delays in sampling. Mechanical degradation due to the harsh environment where the sensors were placed is the most plausible explanation of the degradation, which is further supported by the increasing gradient which is a common phenomenon in faults. Two counterpoints to this are that due to the limited data, the shape of the data should not necessarily be trusted, and the fact that the similarities of degradation and patterns of the faults between the two sensors appears unlikely to happen by chance. This might instead suggest that a part further down the line in the system might be causing the issue, which could affect both sensors simultaneously. Comparison to data of different types of accelerometer sensor degradation could help

determine whether sensor degradation is at fault.

This also proves the importance of continuous checks on data produced. In a system where sensors faults could happen at any time, as well as suddenly, not considering the possibility of faulty data could lead to decisions taken on false grounds, which has the potential to cause catastrophic failure. If this were part of an automatic analysis pipeline, a simple check of data integrity as described by Zhao et. al (2007) according to (1) could detect similar errors in the data, after which the system could more intelligently decide how to consider this.

For future analysis, formula (1) is preferable to determining the degree of faultiness in the data as compared to formula (25), as the latter has weaknesses in how the quantity of sampling delay is grouped within subsections of the data set and does not recognize any delay in data sampling unless $\Delta t$ exceeds 3 ms, both of which (1) handles without issues. In this case, analysis conclusions are unlikely to have changed in any meaningful way because of this.

## 5.2 Discussion and conclusions of data exploration and cleaning of data

The cleaning of the data was successful, with only a minimum of points being out of place of its expected sampling times. Overall, the risk for faults being introduced to the data is small, since no data has been modified and no new data has been introduced to the finished data files. Introducing data from one series when the intention was to introduce data from the other series has likewise negligible effect, seeing that no data changes were identified in the filtration step for the row which was reintroduced, meaning that rows which are reintroduced to the data will inherently always be the same; only the timestamp can be faulty. From the above it can also be concluded that any errors introduced by this process are negligible after considering other sources of errors.

## 5.3 Discussion and conclusions PCA analysis on engine signal data

In the PCA investigations, the anomaly was found not because it was different from the norm, but rather because it was radically different enough within the set of data, taken as a whole, that it accounted for a lot of variance. If PCA is to be used for detecting anomalies, the viability

of which has not been investigated in this thesis, it would have to be done with consideration of what constitutes a normal state, and how to pick out data in a way that the resulting PCA models are comparable.

It was not determined whether the anomaly was a fault or not; it might be possible that whatever happened was a rare, but normal reaction for the system to take during some set of conditions. Consulting experts on the system in question is needed to decide one way or the other.

Talking to experts on the engine it was revealed that some of the variables used in the analysis might not be reliable. These variables also accounted for the most variance in the data, meaning that if they were to be removed, the results are likely to change significantly.

In the case of a section of a run being investigated, the resulting model contained more variance even though there was next to no load change within that period, which suggests that the 15-minute buffer time at the start and end of the run data reduces the variance of the data more than expected. This is likely to be another significant error source in the analysis.

As PCA analysis is highly dependent on the input data provided, the result of PCA analysis of an engine system is more dependent on how an engine is run rather than the actual engine itself. It is therefore critical to take the state of the engine in consideration when doing analysis, as to compare apples to apples rather than apples to pears.

This also showcased a challenge in PCA modelling; the PCA model needs to be updated as state changes happens. Having a single model to apply on, any given state the engine could be in, might work if a model created with data where all states are represented was utilized. However, given the nature of this project, most of all possible states are not represented in available data. An alternative would be to treat each single state individually, so that each state has its own custom-made model.

Kansal (2020) describes the assumptions of PCA. The purpose of these assumptions is likely to be to ensure that analysis results are valid. Ensuring that the data from the system fulfil these assumptions could help in increasing confidence in the analysis results. If similar models were to be achieved in different states, this too would increase confidence in the application of PCA

on data from the system, but it does not guarantee that new states in the system would produce completely different PCA models, which would be incomparable to previous models.

## 5.4 Discussion of the model for categorizing state

State in the model faces the same difficulty as state in partially observed Markov models, and is quite likely to benefit from many of the same techniques used by it. in fact, the three problems as described by Ramage (2007) of likelihood, decoding, and learning, is very interesting for the purpose of learning how the system functions. For this to be possible, however, the initial goal of the model is to make an empirical model for a state, after which is possible to differentiate that state from another state, and then repeating for every state until an understanding of states in the system is received. This can in turn be used to evaluate analysis results, tuning metrics to better differentiate states etc. Once states can be differentiated, the resulting states can either be separated into categorical states, or the data itself, which contains the information regarding the current state, can be fed into, e.g., a neural network (since the system, with good enough metrics, should contain enough information regarding the dynamics of the system to determine state in every single data point).

### 5.4.1 Challenges of the model for categorizing state

There are major challenges in implementing a model for categorizing state. Considering that the system is a control system based on signals from a digital system rather than categorical data, the number of possible combinations from the input and output variables will be too great to handle on a case-by-case basis. One way to handle this is to reduce the number of input states by defining a set of ranges, though this can still become unmanageable. Defining how inputs and outputs should correspond to each other using functions can handle magnitudes greater complexity in comparison. Making the data categorical by using clustering or a similar method might also be possible to achieve the same goal.

The observable state (i.e., the result of the measured values) for a set of inputs is not the whole truth – even if the system is deterministic, it might be that the observable state does not contain enough information to determine the state fully (as in a hidden Markov model). Conversely,

when dealing with a categorical state, it is possible that more than one observable state corresponds to the same state (or considering the output values are on a spectrum, more than one continuous range). That is of course entirely dependent on how that categorical state is defined.

Finding input variables which would account for the entirety of the dynamics in the system is a significant challenge – failing to do so would be equivalent to not taking into consideration gradual changes which occur e.g., after a load change, which changes the characteristics of the system significantly. Metrics could help in this by, for instance, using an infinite impulse response filter to summarize the recent history of the variable, which could provide a more complete picture of the state of the system.

Depending on the system there might be difficulty in defining operational and environmental variables; for instance, the control system might in some cases be made to override inputs made by the user, or might even routinely do so, and environmental variables which mostly are affected by outside factors might still to some capacity be affected by the system itself.

Seeing that it is not always clear how a system work, some measured variables might have little effect on the system, or some unmeasured variables might be very important to determine the state of the system. Aside from consulting experts, the only way to determine this being the case is with analysis. However, analysis can only say which variables appears not to affect the system in the given data set, not what variables are missing.

Another major issue, as mentioned previously in chapter 2.5, is sensor faults. If sensors give out wrong readings it will create bad data, which in turn can lead to incorrect results. There are ways to catch some types of sensor faults, but it remains major challenge. This is, however, an issue in virtually every modeling task.

# 6 CONCLUSIONS AND RECOMMENDATIONS

This thesis was a learning project. Other than providing the theoretical basis for work done in the project as well as in potential future projects, it also was an exploration of methods that could possibly be used in future work. Although the methods described in the theory part of this thesis could be used in future work done by the internal project team, what methods will be used depends on where the focus of future work is.

The specific cases that happened during the project provided ample practice in the use of the tools of data analysis, in this case Python, Pandas, and matplotlib. The case of PCA analysis of engine data was mostly exploratory, but in retrospect, to make any sort of definite statements regarding the data, needs a clearer idea of what is to be explored so that the design of the analysis can be done accordingly, particularly regarding which data to choose for analysis and why.

The model for categorizing state is of particular interest in future work, especially if models using only the momentary state of the engine is to be used in the future. It provides advantages in visualization, making for more accurate comparisons of data even when the system has not been operated in an identical manner, and opens many avenues for analysis. Although there is no guarantee that the model will work for the system it will be applied to, it will provide further insight into how the system works, regardless of any further practical use of the results not being viable.

# SVENSK SAMMANFATTNING

## Datapipor och edgeanalys i motorinstallationer

Intresset för datainsamling och analys har ökat i samma takt som processorkraft har ökat och nya maskininlärningsalgoritmer har blivit bättre, en trend som kallas ”the internet of things.” Molntjänster används ofta för att analysera massiva mängder data i och med dess enorma processorkapacitet, men är inte nödvändigtvis en fungerande lösning i avsides belägna områden med begränsad eller ingen internetkoppling. En s.k. edgeanalys, d.v.s. att analysera data på platsen där den skapats, är ett möjligt alternativ till detta.

Edgeprojektet är ett forskningsprojekt delvis finansierat med offentliga medel, med flera universitet och företag involverade. Detta diplomarbete baserar sig på arbete utfört under projektet med syftet att undersöka edgeanalys i motorinstallationer.

Målet med detta diplomarbete är att undersöka den teoretiska grunden för edgeanalys och analysmetoder som användes under projektet. Dataflödet från de signaler som kommer från sensorer, överförs till datalagring, datarengöring, dataanalys och rapportering av data beskrivs. Relevanta verktyg, metoder och teknologier för dessa steg beskrivs också. Vissa koncept och metoder som är relevanta för analysarbete under projektet undersökts också. Vissa specifika fall som uppkom under projektets gång beskrivs samt resultat från dessa.

Molnalys är ett relativt nytt koncept och som till synes är bättre än edgeanalys med sin enorma processorkapacitet och möjlighet till ekonomisk nytta i och med att man kan dela upp och ge användare exakt den processorkapacitet som de behöver när de begär det. Molnlösningar kräver dock en tillräckligt bra internetuppkoppling, vilket blir en flaskhals om datamängden är för stor. Alternativt fungerar den inte alls om källan data skickas från inte har en internetuppkoppling. Edgeanalys kan hjälpa genom att antingen skicka endast en del av alla data till molnet och analysera resten lokalt, endast skicka aggregerade data till molnet (ta bort kommatecknet) eller genom att hantera alla data endast lokalt.

En av utmaningarna med edgeanalys är att edgeanalys i majoriteteten av alla fall bör fungera autonomt. Detta betyder att om edgesystemet är en kritisk del av ett större system, till exempel om det är en del av eller används direkt av ett automationssystem, finns krav för att försäkra

att fel kan hanteras på ett sätt som förhindrar allvarliga olyckor.

Maskinslitage uppkommer långsamt i början, vartefter takten av slitage ökar exponentiellt. Ett relaterat koncept är sensordrift eller tendensen att sensorer stegvis ger felaktiga resultat. Att upptäcka och särskilja dessa problem är ett utmanande problem, där den bästa lösningen är att jämföra med flera liknande sensorer.

Markov-system, d.v.s. system med Markov-egenskapen, är system vars framtida lägen kan bestämmas utgående från dess nuvarande läge utan att ta i beaktande andra historiska data. Markov-modeller kan delas in i om läget som systemet befinner sig i är känt eller inte samt om det är kontrollerat av en agent eller inte.

Principalkomponentsanalys (engelska, Principal Component Analysis, PCA) är en dimensionsreduktionsmetod. Metoden baserar sig på att finna nya axlar för variabler (som utgör dimensioner) som sammanfattar data i färre antal nya dimensioner medan så lite information som möjligt går förlorad. Dessa nya axlar kallas principalkomponenter och väljs ut genom att hitta den axel som har största möjliga variation, och sedan välja näst största axel med största möjliga variation som har en 90 graders vinkel med alla redan existerande principalkomponenter. PCA har fördelen att kunna sammanfatta data med många olika variabler i ett betydligt mindre antal principalkomponenter, men med nackdelen att dessa principalkomponenter inte har någon förutbestämd betydelse.

Ett schema för dataanalys med syftet att skapa algoritmer för specifika analysändamål har tagits fram i detta diplomarbete. Schemat går igenom allt från att utforska data till att testa hypoteser och utveckla en algoritm som bevis för att algoritmen fungerar i praktiken.

För att avgöra om data från accelerometer-sensorer fästa på en motor var dugliga för ändamålet undersöktes dess tidsstämpeldata i syftet att se om dess sampeltid var tillräckligt snabb och konsekvent. Detta skedde genom att skapa en dataanalyspipa med hjälp av Python-biblioteket Pandas.

Preliminära resultat visade att efter att vissa initiala problem löstes var data från sensorerna tillräckligt bra för användning i analys. En senare undersökning visade att detta inte stämde för senare set av data. Ytterligare undersökningar visade vad som verkade vara en trend av

försämrad prestanda. Inga slutgiltiga svar varför detta skedde hittades, men bland de troligaste orsakerna är att sensorerna tog skada av miljön på motorerna.

Rådata från en motorinstallation så som den var lagrad i en databas var inte i en form som var användbar för dataanalys. För att använda dessa data i analyssyfte behövdes en databehandlingspipa.

Databehandlingspipan delades in i olika steg; separation av data i olika motorkörningar, separation av variabler i olika grupper, bortfiltrering av datarader med irrelevanta data, borttagning av intervall med för hög sampelhastighet och återintroduktion av korrekta datarader.

För att separera data i motorkörningar användes sensorsignaler vars värde är noll om motorn är avstängd och mer än noll om motorn är på. Med hjälp av detta kunde olika körningsrundor identifieras. Systemet tog sampel på två olika grupper av variabler separat, men buntade ihop dessa grupper på ett sätt som skapade redundanta data. Med hjälp av grundliga undersökningar identifierades dessa två serier exakt. Genom att se på när ändringar skett från punkt till punkt kunde en rad data tilldelas antingen den ena serien eller den andra.

I data förekom även korta tidsperioder där sampelfrekvensen var betydligt högre än normalt, vilket konstaterades introducera redundanta data. Dessa rättades till med att ta bort alla rader förutom den sista i tidsperioden.

I steget där gruppen av variabler separeras kunde det i sällsynta fall hända att vissa rader av data togs bort av misstag. Detta korrigerades igenom att återintroduceras de försvunna raderna som det sista steget.

Efter att data hade behandlats enligt stegen ovan separerades data i separata CSV-filer för analys. Datakvaliteten var god efter behandlingen, där varje separat körning i en majoritet av fallen hade mindre än fem perioder med uppehåll i sampling och varje uppehåll inte var signifikant lång.

Principalkomponentsanalys undersöktes som metod på motordata, dels som metod att utforska data, dels i avsikt att se om det kan användas som en del av senare analysarbete. 31 olika

körningar undersöktes ytligt, där vissa arbiträrt utvalda körningar undersöktes grundligare.

Majoriteten av all variation fanns i de fyra första principalkomponenterna, där PCA-modeller baserade på data från individuella körningar i allmänhet hade mer variation förklarad i de fyra första principalkomponenterna jämfört med en modell som baserades på alla data. Den första principalkomponenten verkar vara starkt förknippad med variabler som hör ihop med motorns last. I de scorefigurer som skapades var det även tydligt att motorn stabiliseras med tiden på olika lastnivåer.

Under analysarbetet noterades även hur systemet ändrade sig mitt i datasetet på (datasetet på) hur score-figuren av modellen gjord på alla data hade två distinkta punktsvärmar på principalkomponent två. Orsaken till detta var rutinunderhåll, och de variabler som hörde ihop med principalkomponent två visade sig ha ändrats i samband med detta underhåll. Dessutom upptäcktes (ta bort ordet) en anomali i en av körningarna genom en PCA-scorefigur, och källan till denna anomali identifierades.

Det visade sig att PCA har potential som en metod för att både utforska och analysera, men med vissa utmaningar. Även om en anomali upptäckes med PCA så är PCA inte en metod med vilken man detekterar anomalier. PCA letar efter de största källorna av variation, och är endast ett sätt bland många att undersöka data.

För det andra är PCA som metod känslig för dess indata. Eftersom den största källan till variation i motorn är lasten och hur motorn körs i allmänhet, är analys av motordata som inte tar hänsyn till läget i motorn i det närmaste oanvändbar; motorns läge behöver tas i beaktande.

En av de större utmaningarna inom dataanalysen i allmänhet var komplexiteten i de data som användes. Med 113 olika variabler behövs någon sorts metod att klassificera läget i motorn. En metod för hur detta kunde uppnås har tagits fram i detta diplomarbete.

Målet med detta diplomarbete har varit att undersöka lämpliga metoder för edgeanalys i motorinstallationer. Det utgör ett delarbete i ett större projekt och fungerar som en grund för projektgruppens fortsatta arbete.

Om man i fortsatta studier använder sig av metoder som är beroende av läge rekommenderas

ytterligare undersökningar i hur läget i ett system kan representeras samt undersökningar om lägena i fråga är användbara för att identifiera fysiska förändringar i systemet. Även ett flertal andra metoder som presenterades i arbetet visade sig ha potential för vidare arbete.

# REFERENCES

Batewela, S., 2019. *Towards Reliable and Low-Latency Vehicular Edge Computing Networks.* [Masters thesis]

EliteDataScience.com, 2016. *Data Cleaning.* [Online] Available at: https://elitedatascience.com/data-cleaning [Accessed 1 July 2020].

Englund, E., 2018. *Lecture notes "Industriella automationssystem".*

Erik Olsson, P. F. N. X., 2004. Fault Diagnosis in Industry Using Sensor Readings. *Journal of Intelligent & Fuzzy Systems,* Issue 15, pp. 41-46.

Fisher, R., 1936. *Iris Flower Dataset | Kaggle.* [Online] Available at: https://www.kaggle.com/arshid/iris-flower-dataset/data [Accessed 20 July 2020].

Fowler, M., 2006. *Continuous Integration.* [Online] Available at: https://www.martinfowler.com/articles/continuousIntegration.html [Accessed 6 July 2020].

Google, 2020. *What is Clustering?.* [Online] Available at: https://developers.google.com/machine-learning/clustering/overview [Accessed 24 March 2021].

Jaadi, Z., 2019. *A Step by Step Explanation of Principal Component Analysis.* [Online] Available at: https://builtin.com/data-science/step-step-explanation-principal-component-analysis [Accessed 20 May 2020].

Jagidar, S., 2012. Cloud Computing Basics. *International Journal of Advanced Research in Computer and Communication Engineering,* 1(5), pp. 343-347.

Janet E Joy, E. E. P. a. D. B. P. ed., 2005. Appendix C: ROC Analysis: Key Statistical Tool for Evaluating Detection Technologies. In: *Saving Women's Lives: Strategies for Improving Breast Cancer Detection and Diagnosis..* Washington: National Academies Press, pp. 314-321.

Janzen, D. & Saiedian, H., 2005. Test-driven development concepts, taxonomy, and future direction. *Computer*, 19 September, pp. 43-50.

Kansal, M., 2020. *Understanding Principal Component Analysis and their Applications.* [Online]
Available at: https://www.mygreatlearning.com/blog/understanding-principal-component-analysis/#applicationofPCA
[Accessed 13 June 2021].

Kuo, F. Y. & Sloan, I. H., no date. Lifting the Curse of Dimensionality. *Notices of the AMS,* 52(11), pp. 1320-1328.

Lobo, J., 2018. *Test Automation from Simulation to Real-Time Testing.* [Online] Available at: https://de.mathworks.com/company/newsletters/articles/test-automation-from-simulation-to-real-time-testing.html
[Accessed 11 June 2021].

Metz, C. E., 1978. Basic Principles of ROC Analysis. *Seminars in Nuclear Medicine,* VIII(4), pp. 283-298.

Nakamura, J., 2007. *Predicting Time-to-Failure of Industrial Machines with Temporal Data.* University of Washington.

Nicholson, C., no date. *A Beginner's Guide to Eigenvectors, Eigenvalues, PCA, Covariance and Entropy.* [Online]
Available at: https://pathmind.com/wiki/eigenvector
[Accessed 21 May 2020].

Premsankar, G., Francesco, M. D. & Taleb, T., 2018. Edge Computing for the Internet of

Things: A Case Study. *IEEE Internet of Things Journal ,* 5(2), pp. 1275 - 1284.

Ramage, D., 2007. *Hidden Markov Models Fundamentals.* [Online] Available at: http://cs229.stanford.edu/section/cs229-hmm.pdf [Accessed 18 March 2021].

Sarkar, D., Bali, R. & Sharma, T., 2018. Machine Learning Basics. In: *Practical Machine Learning with Python.* Berkeley, CA: Apress, pp. 3-65.

Schuh, W. C., 2003. *Sensor drift compensation by lot.* United States of America, Patent No. US6850859B1.

Sharma, S. K. & Wang, X., 2020. Towards Massive Machine Type Communications in Ultra-Dense Cellular IoT Networks: Current Issues and Machine Learning-Assisted Solutions. *IEEE Communications Surveys & Tutorials,* 8 August, 22(1), pp. 426-471.

Soni, D., 2018. *Introduction to Markov Chains.* [Online] Available at: https://towardsdatascience.com/introduction-to-markov-chains-50da3645a50d [Accessed 18 March 2021].

Spaan, M. T. J., 2012. Partially Observable Markov Decision Processes. In: M. v. O. Marco Wiering, ed. *Reinforcement Learning: State of the Art.* Berlin: Springer-Verlag, pp. 387-414. [Draft]

Sutton, R. S. & Barto, A. G., 2018. *Reinforcement Learning: An Introduction.* 2nd ed. Cambridge, Massachusetts; London, England: The MIT Press.

Taskesen, E., 2020. *pca 1.3.0.* [Online] Available at: https://pypi.org/project/pca/ [Accessed 25 January 2021].

Tharwat, A., 2016. Principal component analysis - a tutorial. *Int. J. Applied Pattern recognition,* 3(3), pp. 197-239.

Verma, A., 2018. *The Relationship between IoT, Big Data, and Cloud Computing.* [Online] Available at: https://www.whizlabs.com/blog/relationship-between-iot-big-data-cloud-computing/
[Accessed 19 April 2020].

Wicklin, R., 2019. *How to interpret graphs in a principal component analysis.* [Online] Available at: https://blogs.sas.com/content/iml/2019/11/04/interpret-graphs-principal-components.html
[Accessed 27 May 2021].

William L. Dunn, J. K. S., 2011. *Exploring Monte Carlo Methods.* Amsterdam; Boston: Elsevier.

Zhao, Y. et al., 2017. Fault Prediction and Diagnosis of Wind Turbine Generators Using SCADA Data. *Energies,* 10(1210), pp. 1-17.