LAPPEENRANTA-LAHTI UNIVERSITY OF TECHNOLOGY LUT
LUT School of Energy Systems
LUT Electrical Engineering

Oskari Kauppinen

MACHINE LEARNING WITHIN INDUSTRIAL POWER DISTRIBUTION

Examiners: Professor Jero Ahola
D. Sc.(Tech.) Antti Kosonen
Supervisors: Juha Alamäki
Professor Jero Ahola

ABSTRACT

Lappeenranta-Lahti University of Technology LUT
LUT School of Energy Systems
LUT Electrical Engineering

Oskari Kauppinen

Machine Learning Within Industrial Power Distribution

Master's thesis

2019

52 pages, 16 figures and 8 tables

Examiners: Professor Jero Ahola
D. Sc.(Tech.) Antti Kosonen

Supervisors: Juha Alamäki
Professor Jero Ahola

Keywords: Machine Learning, Neural Network, LSTM, Circuit Breaker, IOT, Predictive Maintenance, Fault Detection, LUT

In the age when data collection and data processing is ever cheaper and more powerful, implementing machine learning algorithms are becoming more and more prevalent. Also the potential gains are becoming more and more obvious. This is why studying these opportunities are more and more necessary and topical at the moment. This thesis researches different fault analysis method done with machine learning algorithms on the basis of industrial time series data. The thesis presents all the devices that will be used for the analysis and their usual faults. After that thesis goes through different methods of machine learning. An algorithm that best fits the data being used will be chosen.

In this thesis there is circuit breaker data from ABB's Emax 2 circuit breakers that will be used for this thesis. Most common problems in circuit breakers are jamming of mechanical switching components. For the analysis current signal from the circuit breakers are being be chosen. The chosen algorithm for the application of this thesis is the Greenhouse system. The algorithm developed for the thesis has a Long Short-Term Memory algorithm. That is used to predict current signal values. These predicted values are used to compare them between the actual values and a prediction error is calculated based on the difference of those. After that error values are being normalized and a boundary is defined to differentiate the faulty signals from healthy ones.

Thesis tests two different styles of current signals. Based on the tests can be deducted that the algorithm can predict normal behavior of a stable signal well. In the case of a signal that has really hard cyclic change of values, the algorithm quite often marks normal behavior as faulty. With future development the algorithm can be developed into a working part of fault detection and even as part of predictive maintenance.

TIIVISTELMÄ

Lappeenrannan–Lahden teknillinen yliopisto LUT
LUT Energiajärjestelmät
LUT Sähkötekniikka

Oskari Kauppinen

Koneoppiminen teollisuussähköverkoissa

Diplomityö

2019

52 sivua, 16 kuvaa ja 8 taulukkoa

Tarkastajat: Profesori Jero Ahola
TkT Antti Kosonen

Ohjaajat: Juha Alamäki
Profesori Jero Ahola

Hakusanat: Koneoppiminen, Neuroverkko, LSTM, Katkaisija, IOT, Ennustava huolto, Vian etsintä, LUT

Tietoliikenteen ja sen tiedonprosessoinnin halventuessa ja kehittyessä jatkuvasti alkaa koneoppimisalgoritmien sovellus tulla ajankohtaisemmaksi, ja myös niillä saavutettava potentiaalisen hyödyn arvo alkaa tulla enemmän ja enemmän esille. Täten asian tutkinta on hyvinkin tarpeellista ja ajankohtaista. Tämän diplomintyö tutkii erilaisia koneoppimisen algoritmeja teollisuuden aikasarjadatan pohjalta tehtävään vika-analysointiin. Työ esittelee sekä analysoitavat laitteet ja niiden yleisimmät viat. Sen jälkeen työssä käydään läpi erilaisia koneoppimisen menetelmiä, joiden pohjalta selvitetään parhaiten käytössä olevalle datalle soveltuva algoritmi.

Tässä työssä on käytössä katkaisijadataa ABB Emax 2 katkaisijoilta. Katkaisijoiden pääasiallisina vikoina ovat mekaaniset jumiutumiset. Katkaisijadatasta tutkitaan virtasignaalia. Työtä varten valitaan Greenhouse järjestelmä toteutukseen. Toteutettava algoritmi sisältää Long Short-Term Memory -algoritmin, joka ennustaa virtasignaalin arvoja. Näitä ennustustettuja arvoja verrataan tapahtuneisiin signaalin arvoihin ja näiden pohjalta lasketaan ennustuksen virhe. Virhettä sen jälkeen normalisoidaan ja määritellään raja, jossa signaaliarvot luetaan vioiksi.

Työssä testaan kahta erityylistä virtasignaalia. Testien pohjalta voidaan päätellä, että algoritmi kykenee hyvin ennustamaan stabiilisen signaalin toiminnan. Signaalin kohdalla, jossa on hyvin vahvaa syklistä käyttäytymistä, algoritmi helposti merkitsee normaalin käyttäytymisen ääritilanteissa virheeksi. Jatkokehittämisellä algoritmista voidaan saada toimiva osa viantunnistusta ja mahdollisuus jopa osa ennakoivaa kunnonvalvontaa.

PREFACE

This work is done for ABB as a part of the process for a future with more machine learning based systems helping to minimize problems with devices. Thanks to Juha Alamäki who has been helping me lay the basis for this thesis and just by being supportive all around from ABB side. Thanks to Jero Ahola who agreed to oversee and give some professional opinions from the university side. Special thanks to my girlfriend Milla for being there to listen and play around with ideas.

This has been maybe the most interesting projects that I've been involved in. Really challenging, but interesting. Also maybe one of those few projects that I've actually finished. The whole process of learning a whole new field for this thesis has taken a lot of time, but has also given me a new career path. For that reason I'm really grateful to be able to do a master's thesis that was mostly one of my wildest dreams before it sits before my eyes now.

Oskari Kauppinen
In Helsinki 19.06.2019

TABLE OF CONTENTS

Abbreviations

AD	Anomalous Detection
AWS	Amazon Web Services
CB	Circuit Breaker
CPU	Central Processing Unit
GPU	Graphics Processing Unit
IT	Information Technology
KPI	Key Performance Indicator
LSTM	Long Short-Term Memory
M-distances	Mahalanobis distances (Mahalanobis 1936)
ML	Machine Learning
NN	Neural Network
OEM	Original equipment manufacturer
PM	Predictive Maintenance
RNN	Recurrent Neural Network
SF6	Sulfur hexafluoride. A gas used in many breakers
SML	Supervised Machine Learning
SQL	Structured Query Language
USML	Unsupervised Machine Learning

List of Symbols

Γ	Gate Function
$\hat{y}$	Prediction
μ	Mean
ω	Coefficient
$\phi(x)$	Basis Function
Σ	Covariance Matrix
$\tilde{c}$	Candidate for Updated Cell in LSTM
Ξ	Input Data
a	Output in LSTM
b	Bias Vector
c	Cell State
D_M	Mahalanobis Distance
E	Mean Function
$h(x)$	Hypothesis
L	Loss Function
M	Order of Polynomial
N	Size of Matrix
p	Prediction
t	Time
v	Value
W	Weight Matrix
x	Input Variable
y	Output Variable
A	Ampere
d	Day
h	Hour
Hz	Hertz
m	Minute
S	Sample
s	Second

V	Volt
Var	Reactive Power
W	Active Power

1 INTRODUCTION

This section introduces the thesis first opening the motivation for the subject. This section describes the aims of the work and structure of it.

1.1 Research background

We live in the time where first time applications are becoming really affordable, powerful as seen in (AI Impacts 2015) and also really reliable. This has created the possibility to apply it to new areas where there were once limitations. In the case of this thesis collection of data has become really cheap on the level that it's not a limiting factor anymore. For example IBM suggests that 90 % of data has been created in the past two years in 2017 (Cloud 2017). This is testament by itself about the size of the change. Also computing power is readily available and mostly affordable since the introduction of cloud. No need to buy computers. For example Amazon's pricing is usage based so setting up the system costs nothing and scaling up doesn't require investment in the initial scaling up process (Amazon n.d.). Only the upkeep and data processing themselves add costs (Amazon n.d.). Anymore there is no need to spend huge amounts of money to build up server farms. It only takes a subscription to a cloud and there is huge amounts of computing power at anyone's fingertips.

On the basis of data collection and processing becoming cheap, we come to the reason for this research. At this moment maintenance at factories is mainly is periodic and maintaining happening mostly someone physically going to the device. Now we have the possibility to actually get all the same condition and measurement data in digital form all around the world that we would have next to the device itself.

This opens up the possibility to first move towards a faster reacting maintenance. After collecting the condition and measurement data from a huge number of devices there start to be repeating faults and breakdowns. The amount of data is far too big for anyone to analyze by themselves, so it opens up the possibility to use learning algorithms that see those anomalies and learn from those. After learning those algorithms can be used to predict future occurrences of those anomalies and maintenance can be done before the fault or breakdown actually happens in a controlled manner.

Production facilities like paper mills and bio product mills try to operate at high capacity running

24/7 and only shutting down the mill for maintenance. Sometimes a critical part of a mill breaks down or faults critically and at that point that facility might need to shut down for maintenance. The costs are really high when that happens since it can take from a few hours to days to get everything running at high capacity and quality again.

Algorithms themselves take away unnecessary scheduled maintenance saving money for the owner of those devices. Also the predicting the faults and breakdowns in the future makes it far more easier to prevent unscheduled shutdowns of production facilities by maintaining or replacing the devices in a scheduled shutdown. So these predictive maintenance methods can create possibly even large savings.

This thesis is done for ABB which is as a company realizing these possibilities and investing hugely for the development in the area. This work is heavily related to the platform ABB Ability Electrification and the data acquired by the platform. This thesis and its findings will hopefully in the future integrate to the ecosystem of the said platform.

1.2 Objectives and limitations

This thesis aims to create understanding around machine learning in the context of industrial power distribution data and how to create more automatic algorithms for faults that possibly can be used in the future to create algorithms that helps Predictive Maintenance (PM) operations. creating a base and good principles to choose Machine Learning (ML) algorithms for this kind of data. At this point the data is really raw, mostly unhandled and there isn't unlimited amount of it. Starting this thesis assumptions of the quality of the data have to be made. That assumption is that the data used for this thesis is wide enough and can be used for the prediction. Still there might be limitations within the data that is needed to work around.

In this thesis circuit breaker data is being analyzed with a ML algorithm to find faults. Different methods of ML will be analyzed and the best method based on the limitations in the data and usual faults will be chosen. Need is for a model that can help in analyzing which devices need closer attention. This means that the model needs to be as simple as possible to use and can be the most helpful in the everyday upkeep of the devices. A simple trigger for anomalies can help direct resources for analysis on devices that are more inclined for faults. Circuit breaker usual faults are also listed to direct the ML algorithm decision.

Even though in the power distribution system there are many different equipment connected, and there are data collected of them, in this thesis breaker is the equipment that is being studied. This is because these are the main equipment in the power distribution and faults in them might have huge effect on the whole production and maintenance of those is mostly impossible without a shutdown of production.

1.3 Structure

The thesis will begin with an introduction section which will open up the objectives and the structure of the work. It will answer to the questions what is done in the thesis and how and what are the limiting factors for the work.

Next there are two sections that will go through the theory of first the devices in focus and their usual faults and reasons for breakdowns. After the theory part an analysis of risks is formed to decide the most meaningful faults/breakdowns that will be concentrated in the later parts. The other section is focused on the methods of ML describing different main types and their use cases. After this the best methods for this work are chosen for later use.

The fourth section introduces the data that is on hand for model creation. The before decided faults/breakdowns and ML methods are used for building a model around them as long as can be done with the given data. Also in this section bases for future PM are being laid for future development. The fifth section is the discussion section where further discussion of the subject is done closing the thesis' research part. The sixth and last section wraps the whole thesis going through the main points of the work.

2 EQUIPMENT AND THE PROPERTIES OF THEM

2.1 Circuit Breaker

Circuit Breaker (CB) and its reliability is one of the main subject of this thesis. This is why it's important to look into the device and find out how it works and what are the main faults that can happen so that a direction for the later analysis can be taken to that an accurate analysis of the health of the CBs can be made.

A CB is the main component used in power distribution that is used to detect faults with or without relays and disconnecting electric circuit so that the faults won't cause other devices to malfunction or break because of the high currents usually created by the faults. (Leslie Hewitson 2005, p. 70-73)

The main problem in opening loops is the arc created between the contacts when there is high voltage and high current especially in short circuit situation. For this there are a few different ways to go around to minimize the arc caused by opening breaker contacts. There are four main different CB types. Oil, air, and vacuum CBs. The main function of opening contact and creating long enough space between the contact surfaces that the arc created by opening it can be extinguished. (Leslie Hewitson 2005, p. 70-82)

The contact opening/closing mechanism is the main mechanical part of the design of all CB types. This means that at least on the surface that mechanism will be the easiest subject for CB's internal faults.

2.1.1 Emax 2

The breakers used for the analysis of this thesis are ABB's SACE Emax 2 breakers (ABB n.d.). The breakers are modern low voltage air CBs. There are three different product sizes in the acquired data being analyzed. These models are E2.2, E4.2 and E6.2. The communication used to get the data from the devices is Modbus TCP. The signals that have been collected from the breakers are introduced in table 2.1. These signals have been decided to catch the momentary usage of the devices to follow trends. Also the collection of the CBs' internal Key Performance Indicator (KPI)s gives a good idea of the operation cycles and such. These will be the signals used in the analysis part of this thesis.

Table 2.1. Signals collected from Emax 2 breakers and the sampling times of those.

Signal name	Signal unit	Sampling time
Contact wear	%/650	1d
Number of CB operations	N	1d
Number of manual CB operations	N	1d
Number of CB protection trips	N	1d
Global status 1	N	1s
L1 Current	dA	1s
L2 Current	dA	1s
L3 Current	dA	1s
CB serial number	N	1d
CB name	N	1d
Relay serial number	N	1d
Date of CB last maintenance	N	1s
Trip history	N	1s
Event history	N	1s

2.1.2 Circuit breaker faults

Faults in CBs are rare. Mostly the faults are related mostly to breaker not opening or closing due to faulty control wiring, uncharged actuators and by being stuck as is founded in researches (Brown 2002, p. 85-86) and (Lindquist 2008, p. 816). Also dielectric faults cause smaller part of the problems due to breakdowns to earth, CB not able to break the current while opening, breakdown between the contacts and other more minor causes (Janssen 2014, p. 811). CBs can suffer internal faults when it's not able to open or close. These cause the before mentioned dielectric breakdowns and these are differed by the breaker insulation medium. Also erosion of the contact equipment creates fault. Every time the breaker operates, small part of the contact material evaporates (Brown 2002, p. 85-86).

The 2.1 is created from the faults in one high voltage CB from the Swedish and Finnish power distribution networks. The figure shows that the opening and closing lock mechanisms are the main cause for the faults in CBs. Also it shows that there is strong correlation between the number of faults and on the amount of operations of the CBs per year. Most of the faults are caused by age, wear and corrosion. This accounts to around 50 % of major faults of CBs (Janssen 2014, p. 810-811). Design, manufacturing and incorrect maintenance related faults account to 15 % of the major faults.

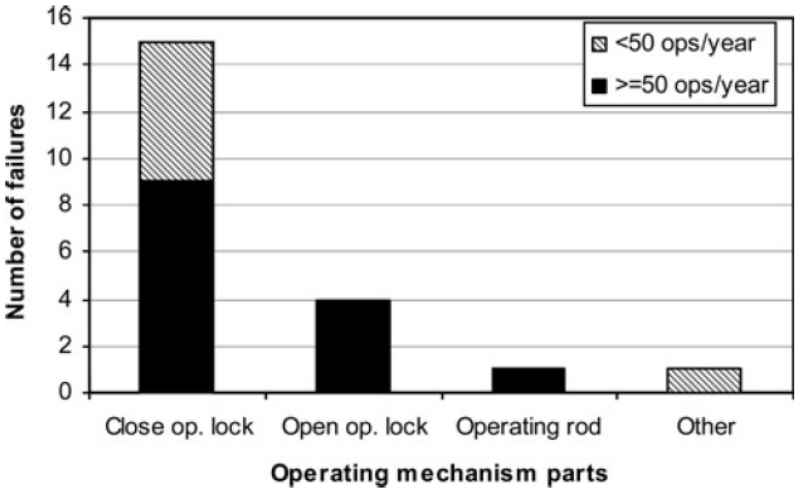

Figure 2.1. Parts failing in the mechanics in a high voltage SF6 breaker (Lindquist 2008, p. 816)

Based on these findings it's clear why analysis around the usage of the CBs and the surroundings of them is really the key for detailed analysis. Also component details of different batches of devices would help since usually there are slight differences in used components in different batches and also between different models of devices. For example there are three different models in the Emax 2 product family.

The data analyzed from each source is collected from high voltage power line CBs so it might not be 100 % accurate when trying to analyze the faults in low voltage CBs that are located in an industrial environment. The main differences though might be the rate of usage which might be much more tasking in industrial environment. That just makes the wear factor on the devices more crucial which is the part that is the focus on this thesis.

Also the environmental factors might be less affecting since the electrical rooms inside the industrial environment are air conditioned. That means the CBs are in a controlled temperature and humidity so the devices should be far less prone to failure compared to the CBs in the study that might be in an unmanned substation without much climate control or are located somewhere along the power lines in outside environment.

2.1.3 Relays used in data analysis

There is another device that is related to the breakers. This is the relay. Relays are connected to the breakers and collect signals related to the breaker. The signals that are collected from

relays are presented in table 2.2. Since these relays are connected to the Emax 2 breakers for data collection, they might have some useful data that can be used later in the data processing.

Table 2.2. Signals collected from all of the relays and the sampling times of those.

Signal name	Signal unit	Sampling time
Frequency	Hz	1s
CB Remaining Life Phase A	d	1s
CB Remaining Life Phase B	d	1s
CB Remaining Life Phase C	d	1s
CB Inactive Time	d	1s
Phase Current A	A	1s
Phase Current B	A	1s
Phase Current C	A	1s
Phase Voltage Phase AB	V	1s
Phase Voltage Phase BC	V	1s
Phase Voltage Phase CA	V	1s
Residual Voltage	V	1s
Power Factor	‰	1s
Accumulated Forward Active Energy	kW	1s
Accumulated Reverse Active Energy	kW	1s
Accumulated Forward Reactive Energy	kVar	1s
Accumulated Reverse Reactive Energy	kVar	1s
Active Power	kW	1s
Reactive Power	kVar	1s
Apparent Power	kVA	1s
Travel Time During Opening	ms	1s
Travel Time During Closing	ms	1s
Charging Time Of The CB Spring	ms	1s
Number of CB operations	N	1s
Breaker state	integer	1s
IRF	integer	1s
Trip Circuit Supervision	integer	1s
Current Circuit Supervision	integer	1s

3 MACHINE LEARNING

This chapter looks at the different methods for machine learning that are used in modern data processing. Later after that this section decides on the best methods to use later in the ML implementation part of this thesis.

3.1 What is Machine Learning

Normal way of programming software is sort of telling the computer what it should do. Machine learning a way to give the computer a way to learn tasks by giving it learning material to learn from (Kubat 2015, p. xi-xiii). Next how it actually does the learning process will be looked more closely.

To understand how the conclusion is made in ML the understanding of the mathematics behind it is crucial. The first part is understanding where to start and where to end up. For any ML algorithm the starting point is input data Ξ which many times called as the training set. In this thesis x marks the separate input variables inside Ξ. x is a vector containing n amount of variables. Inside the matrix Ξ x has dimension m which is the amount of samples of x. The output for a ML algorithm is a hypothesis $h(x)$ which is depended on the type of learning algorithm in use. The relation between each component is depicted in figure 3.1. (Nilsson 1998, p. 5-7)

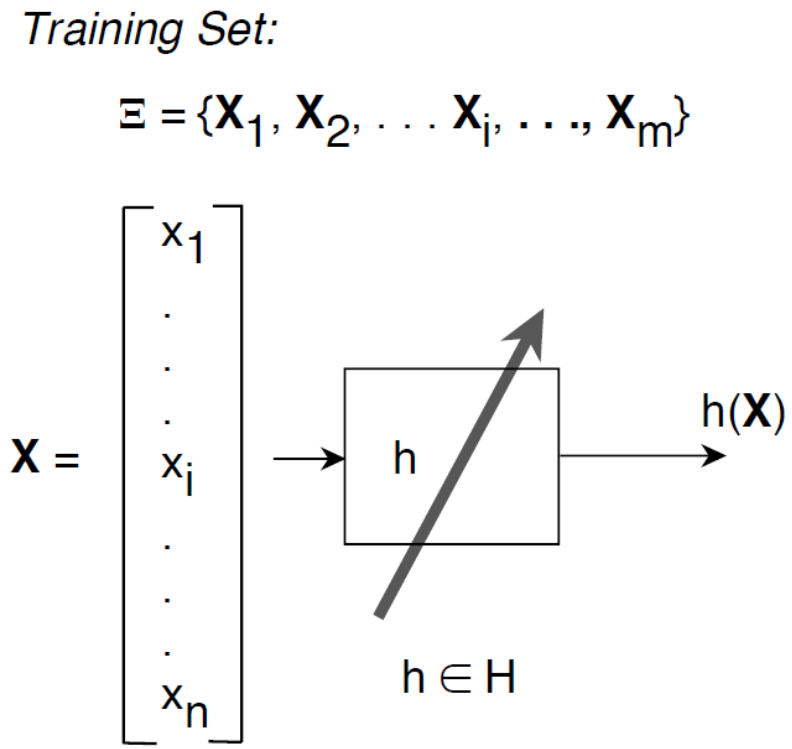

Figure 3.1. Relation of input and output (Nilsson 1998, p. 6).

Between the input and output layers there's level h, which is process level which processes the inputs and outputs the hypothesis $h(x)$. In one of the most simplest forms it can be thought as a two-dimensional function f where a plot curve is is implemented to fit the two-dimensional plane. An example of a fitted curve based on data is found in the figure 3.2. (Nilsson 1998, p. 5-7)

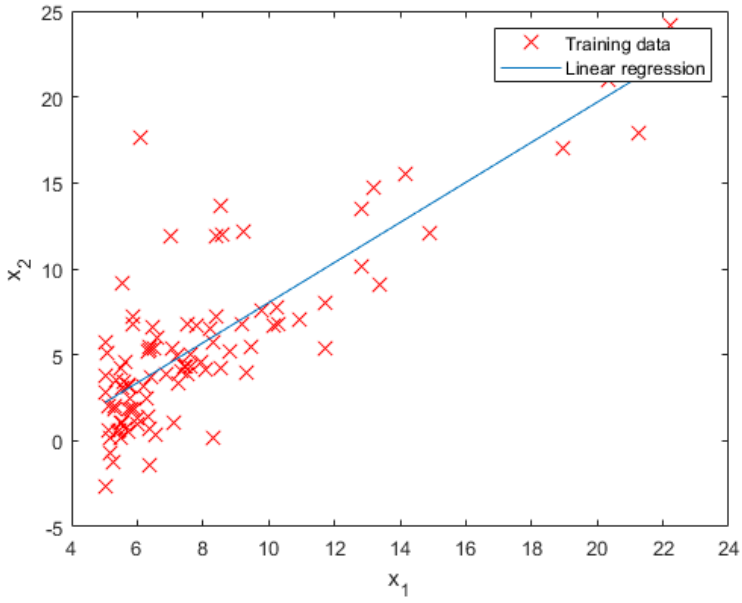

Figure 3.2. Two-dimensional curve fit.

In the next sections two well known fields are being looked closer. Those are Supervised Machine Learning (SML) and Unsupervised Machine Learning (USML). These two ways of ML differ quite a lot and give a really varied tool set to use for the analysis of the empirical data. There is also one other field of ML that won't be looked at. It's called reinforcement learning (Bishop 2006, p. 3-4). This is because it's more suitable for environments, where the use of trial and error can be utilized. For production equipment in an plant and also just for the sake of the cost, this would not be anywhere near feasible.

3.2 Supervised learning

Problems in which you have in your training data that consists of input vectors and their corresponding output vectors is called supervised learning (Bishop 2006). This means that in SML we have a set of outcomes for the algorithm. Example of a supervised learning algorithm is a logistical regression of patterns depicted in 3.3. In the picture we have set of different numbers ranging from 0-9. The algorithm will predict which number does which sub picture represent. In this case we have an input of vector x which consists of the pixels of the written number and its frame. The output for the algorithm is vector y which is a logical vector of 10 depicting the numbers from 0-9.

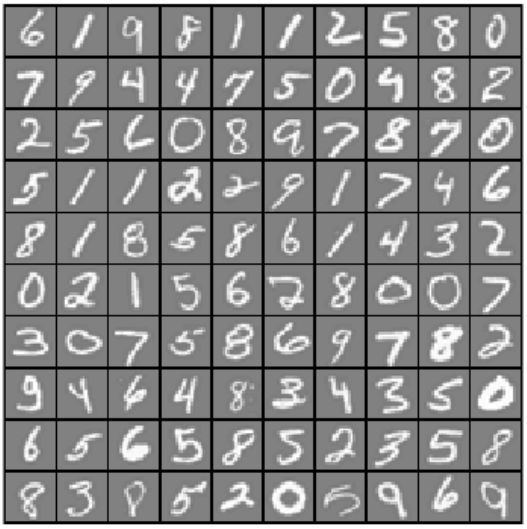

Figure 3.3. Handwritten numbers from 0-9

When there is this set y output value/vector the training material has to have that y vector to steer the ML algorithm towards those answers. In supervised learning there is two main types. Those types depend on the output type. First one is classification problem. In classification there are only a number of fixed outcomes like the number prediction mentioned in the previous paragraph. If the problem has one or more outcomes that might be a range of different values, for example predicting the market price of a building, it's called a regression problem.(Bishop 2006)

In supervised learning, the learning is done by minimization of what is called the cost function J known also as loss function. An example of minimization of J is in the figure 3.4. The cost of the function can be thought as the error between the actual outcome and the predicted outcome. So when the cost of the function reaches zero, the prediction rate reaches 100 %. The output value can be calculated using the formula

$$\hat{y}(x, \omega) = \omega_0 + \omega_1 x + \omega_2 x^2 + \ldots + \omega_M x^M = \sum_{j=0}^{M} \omega_j x^j. \tag{1}$$

where $\hat{y}$ is the prediction based on the values of x and the coefficient ω. In the formula we can adjust the value M which denotes the order of the polynomial and so also the power which x takes. (Bishop 2006)

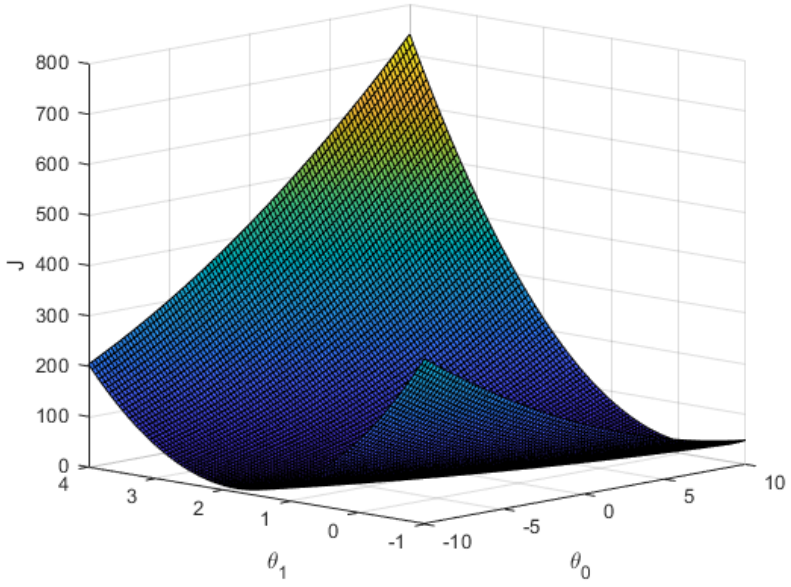

Figure 3.4. An example of minimizing cost J which is dependent on the values of θ_0 and θ_1.

To calculate the cost of the prediction is done by

$$J = \frac{1}{2}\sum_{n=0}^{N}(\hat{y}(x_n, \omega) - y_n)^2. \tag{2}$$

where y_n is the reference output for the equivalent input x_n. With this formula the chosen ω can be tested and seen if it's a good coefficient to be used to get the best prediction of the true output value. the $\frac{1}{2}$ is dependent on the algorithm used to minimize the cost. (Bishop 2006)

Figure 3.5 uses different order of polynomials to create a fit for the function $\sin(2\pi x)$. In really small order polynomials where M is 0 or 1, the fit is too simple to rightly represent the data very well. This underfitting is called bias. If the chosen order of polynomial is too large there comes a problem of overfitting. In the figure 3.5 the order $M = 9$ has noticeably too complex representation. So much so that it makes the prediction worse than a lower polynomial representation. This overfitting is called high variance. (Bishop 2006, p. 6-8)

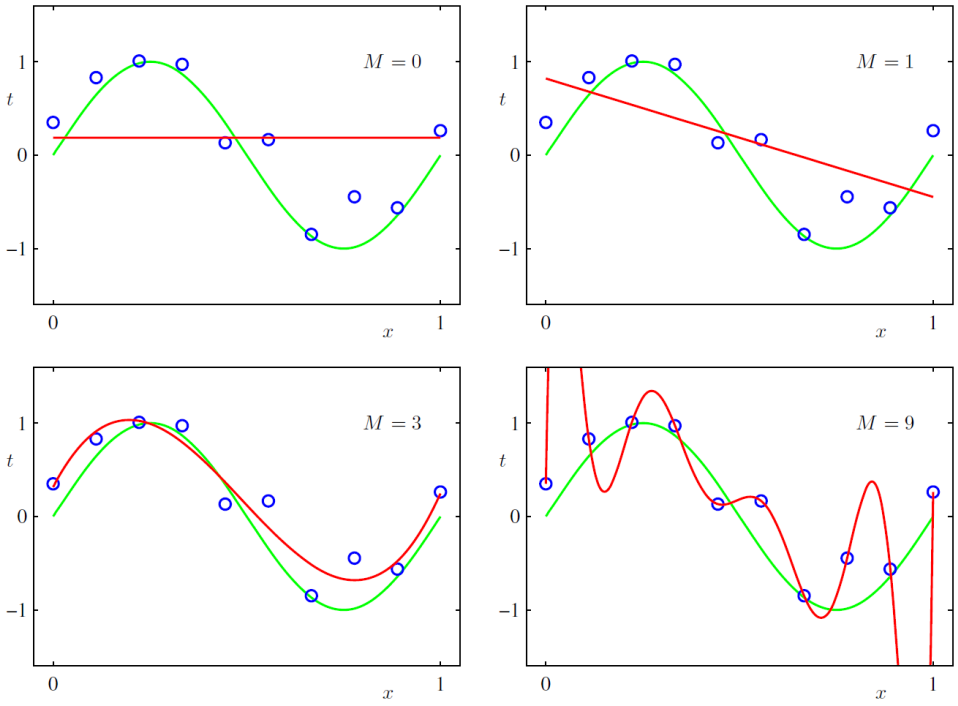

Figure 3.5. Different order polynomial fits M to data points that correspond a $\sin(2\pi x)$ function (Bishop 2006, p. 7).

In this section the basis for SML is laid, but for actual processing of data, an algorithm scheme has to be decided. In the next sections different popular methods around this supervised learning will be inspected more deeply.

3.2.1 Linear Regression

Linear regression tries to give an accurate estimation of one or more continuous y target value from a X-dimensional input vector (Bishop 2006, p. 137-138). One example of this is figure 3.2. In this figure an approximation $\hat{y}$ is done by making a linear fit to the input variables. One more concrete example would be a prediction of housing prices, where input variables might include for example size of house, year it was built, location, yard size, number of bedrooms, etc. These would give an output of an estimation $\hat{y}$ for the price for the house.

The simplest method to get the linear regression prediction $\hat{y}$ from the input matrix X is called linear basis function model. The prediction is done with equation 1, where in this case the ω is the variable that is used to optimize for the prediction model.

ω_0 is called the bias term which is a offset of the data. Usually creating a pure linear representation creates limitations for the fit of the model. That's why it's common to extend the model with

linear combinations of fixed nonlinear functions of x, in the form of

$$\hat{y}(x, \omega) = \sum_{j=0}^{M-1} \omega_j \phi_j(x). \tag{3}$$

where $\phi_j(x)$ is the basis function. Usually $\phi_0(x) = 1$ is used since the bias ω_0 term. (Bishop 2006, p. 138-140)

By using the non-linear functions the prediction $\hat{y}$ has the ability to be truly non-linear. The downside for using these non-linear models is more complicated model. One of the simplest non-linear functions is $\phi_j(x) = x^j$. In total there are a huge variety of different possible functions to use to create non-linear models.

3.2.2 Neural Networks

Before this point the thesis has discussed only about linear algorithms. Linear algorithms can be used really variously, but there are situations that phenomena that is being researched doesn't necessarily have linear elements. For this there is a algorithm called Neural Network (NN). On the surface the NN algorithm looks like any other SML algorithms. It has input layer and an output that can be e.g. a logical classification. This situation is depicted in the figure 3.6. The part that happens inside is what differs the NN from other algorithms completely.

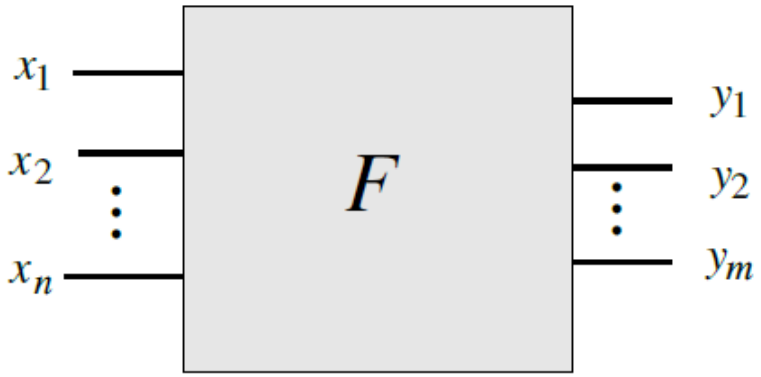

Figure 3.6. A Neural Network "black box" as seen from outside (Rojas 1996, p. 29).

NN is really an artificial neural network that tries to mimic a neuron of the human brain. This is because human brain has a really good ability for learning and those neurons are the main culprit that have been improved by evolution through millions of years (Rojas 1996, p. 3-23). A NN is basically a collection of nodes as seen in 3.7. These on themselves are primitive functions

(Rojas 1996, p. 23). As a collection though these nodes can do a lot. NNs are being used widely in pattern recognition solutions (Bishop 2006, p. 225-227). These include solutions like object recognition from pictures and natural language applications (Goodfellow, Bengio, and Courville 2016, p. 164).

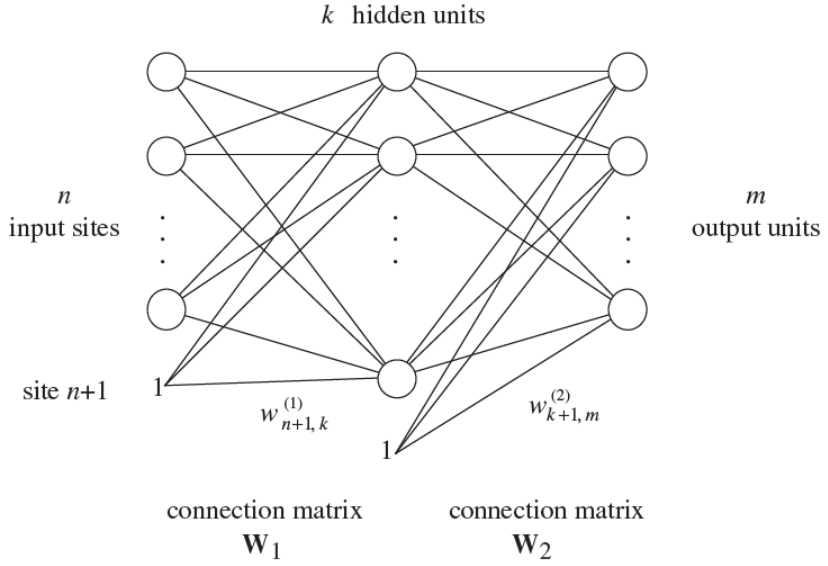

Figure 3.7. A three-layer neural network (Rojas 1996, p. 165).

3.2.3 Feed-forward network

One of the most used neural network algorithms is the feed-forward network. Next it will be inspected to get a sense of the NN. The feed-forward network consists of input sites n, hidden unit(s) k and the output layer m. Figure 3.7 shows a simple three layer NN. These layers are connected with weights W_1 and W_2. There are $n + 1$ weights in W_1. This is due to used bias that is added for convenience reasons. The bias is added to all of the layers other than the output layer. The size and amount of hidden layers can differ depending on the application. (Rojas 1996, p. 55-75)

Feed-forward is called so since the algorithms flow forwards without having any feedback (Goodfellow, Bengio, and Courville 2016, p. 164). Also what sets NN different from other algorithms, is the backpropagation. When a round of the NN is calculated, a backpropagation is calculated. This calculates the whole error to all of the hidden units (Bishop 2006, p. 241-249).

3.2.4 Long Short-Term Memory

There is a different type of NN algorithm type called Recurrent Neural Network (RNN). There has been a lot of development around this type of NN. In this thesis one specific type on RNN will be concentrated and that is Long Short-Term Memory (LSTM). This is because recently there have been many different advances around anomaly/fault detection using LSTM algorithms like (Malhotra et al. 2015), (Lee 2018) and (Bontemps et al. 2017). RNN algorithms answer to the problem of traditional NNs by adding feedback to the model (Connor 1991). For example in natural language ML situation where individual words are being analyzed.

For example in text "Paul is doing laundry" it changes the structure of the sentence, if there are multiple Pauls instead of the one. Conventional NN algorithms don't have relation between the analyzed words. The RNN algorithms on the other hand have. LSTM is a more developed version of the RNN adding a memory element that addresses the vanishing gradient problem of RNN algorithm (Hochreiter 1997).

In the figure 3.8 an usual structure diagram of LSTM can be seen. In the figure X_t means the input for the time T. The time-factor can be thought as a time signal or group of time signals that is moving on every round of iteration moving forward. The iterations are connected to each others (Hochreiter 1997).

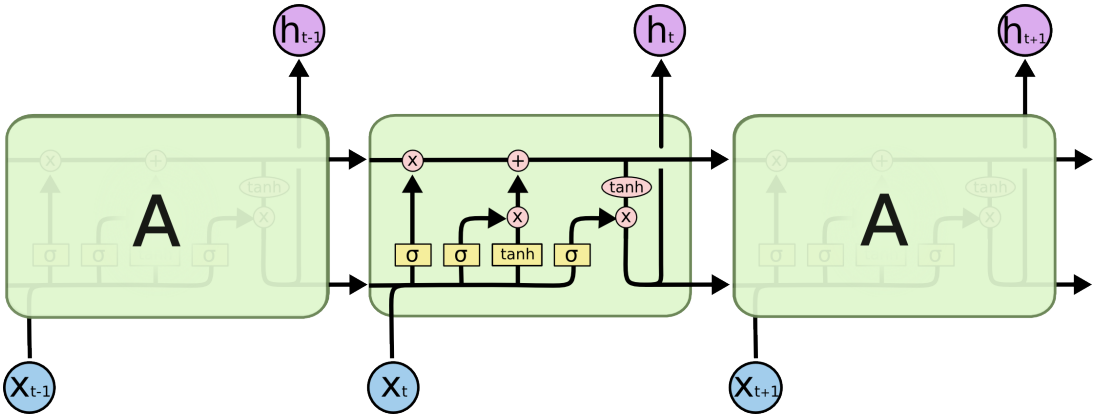

Figure 3.8. A LSTM module with its operations (Olah 2015)

There are a few operations made to the input X_t to determine how much does it affect it has on the hypothesis h_t and on the long and short-term memory by going through σ and $tanh$ gates (Zazo et al. 2016). These gates decide how much of the memory cell is replaced by the input signal. The next equations are needed for processing LSTM taken from (Zazo et al. 2016) and

transformed into more readable form:

$$\textit{Candidate: } \tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c). \tag{4}$$

$$\textit{Update gate: } \Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u). \tag{5}$$

$$\textit{Forget gate: } \Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f). \tag{6}$$

$$\textit{Output gate: } \Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o). \tag{7}$$

$$\textit{Cell state: } c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}. \tag{8}$$

$$\textit{Block output: } a^{<t>} = \Gamma_o * c^{<t>}. \tag{9}$$

where $\tilde{c}^{<t>}$ is a candidate for the updated cell state, W is the weight matrix, $a^{<t-1>}$ is the block output from the last iteration, $x^{<t>}$ is the input values of the current iteration, b is the bias vector, Γ is the gate function, $c^{<t>}$ is the next cell state and $a^{<t>}$ is the block output.

3.2.5 Using LSTM in anomaly detection

Itself LSTM is the same as other SML algorithms so it needs labeled data for the learning process. In the data for this thesis there is no labels for normal or anomalous data. This is because the data is completely raw only including value and a timestamp and there isn't any previous knowledge of malfunctioning devices to be used. This means that it can't be directly used for the needs of this thesis.

However there are though workarounds to be able to use LSTM algorithm for Anomalous Detection (AD) purposes. This solution uses LSTM algorithm to create a model that predicts future values for the devices (Malhotra et al. 2015) and (Lee 2018). This doesn't mean actual

future values, only that the algorithm is given a few subsequent signals and it will predict the signals after that. (Lee 2018) uses three signal values for each LSTM blocks and predicts two future values.

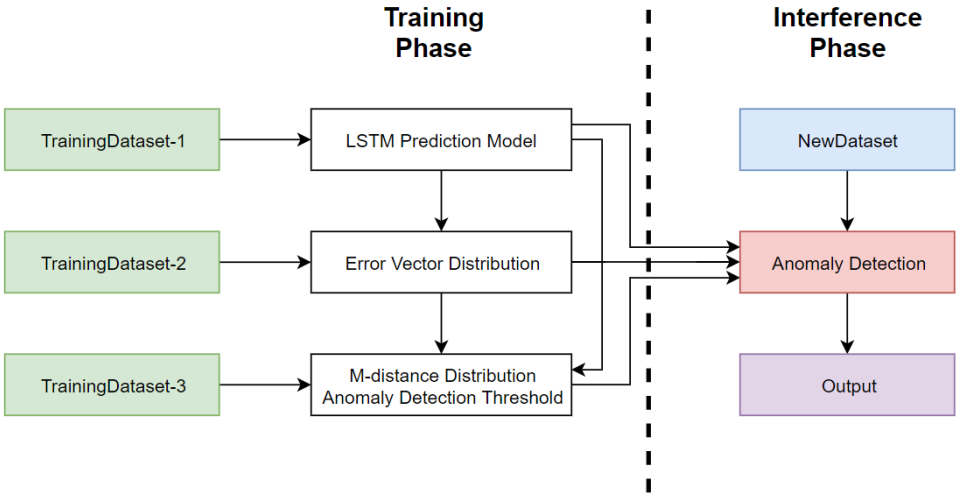

Figure 3.9. The Greenhouse system diagram.

In both (Malhotra et al. 2015) and (Lee 2018) the dataset is split into different collections. The first one is used to train the LSTM model. In (Lee 2018) the next dataset is used to predict future values based on the LSTM model and to compute error vectors based on that. Then the error vectors are being fitted into a multi-variate normal distribution.

The third dataset is used also to predict values and error vectors computed based on the LSTM model. Then Mahalanobis distances (M-distances) are computed based on the error vectors. The resulting distances are fitted to a truncated normal distribution where a user-specific AD threshold is specified. (Lee 2018)

After the this AD model has been created it can be applied on device data to see if there's anomalies. This is done by computing the error vectors of the device data. From this M-distance are computed and the center of multi-variate normal distribution computed. In the end the values which M-distance exceeds given AD threshold are labeled as anomalous.

There are a few different application of different AD models using LSTM, but the Greenhouse system introduced in (Lee 2018) is the only that can be potentially used in this thesis. This is because it's the only algorithm introduced that doesn't need anomalous data to be trained. At this moment in the data at hand, there is no anomalous data.

3.3 Unsupervised learning

In the previous sections SML was introduced. Like SML USML tries to discern patterns. It does it a bit differently. Whereas SML tries to find predictions and categories to predetermined limits, USML tries to group data to discern differences in data. This is called clustering. The amount of clusters can be any amount starting from one. (Bishop 2006)

The most important thing is to decide what is needed to find out of the data. This means that it's good to discern what is the data used for, to be able to create meaningful clusters. For example a t-shirt company trying to decide what measures it uses to create different sizes, can decide to do three different sizes and with that decision create three clusters based on the measures of their customers. Strict decisions don't have to be made since the amount of clusters can be decided based on the efficiency of the clusters.

USML is based on clustering like in the last chapter. Clusters try to separate different groups from each others. These methods include mainly algorithms like K-Means, Gaussian Anomaly detection Next sections will go through more deeply different methods of USML. USML algorithms won't be gone to too deeply in. This is that mostly they are unfitting for the amount of the data at hand. Like the survey (Chandola 2009, p. 20-42) goes through. The algorithms are mostly computationally expensive. And this expensiveness becomes even more cumbersome when talking about the fact that it doesn't use storage able weights like SML algorithms that can be used later when repeating or expanding the data pool.

3.4 Optimizers in ML

In this section optimizers are gone through briefly. This section tries to answer questions what are optimizers and why to use those. In order to find the local or global minimum the most efficiently batch optimization is the best way. These optimizations take a batch of the training set and optimize based on that. In pure gradient descent the optimizations happens for the whole training set so weight update happens only after going through the whole training set. Whereas in batch methods the weight update happens after every batch. The batch size can be as small as one input sample. (Bishop 2006)

In order to avoid giving one batch too much power on the direction of the weights, optimization is needed to direct the learning to the minimum of learning. One used optimizer is Adam. Adam combines the advantages of popular optimizers including AdaGrad and RMSProp (Kingma

2014).

3.5 Choosing right methods for predictive maintenance

At this moment the equipment data that is at hand limits the possible models that can be created. The data is collected only from working equipment. That means that any model that needs anomalous data in order to be used is out of the picture. That mostly removes the possibility to use any kind of supervised learning algorithm. That is since in SML algorithms all types of data is needed to create a model. It is impossible to differentiate a broken equipment if the ML model doesn't know what broken means.

Different to this is the Greenhouse system where because it is created the very point in mind that time series data isn't labeled and also that it might be difficult to come by anomalous data when the devices might work well for many years before any faults. Also USML models can be created where normal operating limits for a device are defined based on some normalization, and devices out of those limits can be found. This means analysis of the equipment data collected and creating limits based on that.

The drawback though is that the models need a lot of manual adjustment to find the right limits of faulty versus normal data. These limits also might need constant adjustment since the devices get older as the time goes on. Also new devices that would be added in the future might have different operating curve which can affect the model. Compared USML to SML the analysis of devices is really intensive in the long run since all the data needs to be analysed throughout to get the results. Whereas in SML the data needs to be run through the algorithm only once.

This is a really meaningful point since the amount of data used in this thesis is already massive and it's only growing. Computation heaviness as a problem is also brought up in (Chandola 2009, p. 19-20) when thinking about a sensor network where the analysis is on online anomaly detection. Online meaning that the analysis has to be made constantly from real time data.

Taking all the points into account, the Greenhouse system is the best direction to take at this point. That's why it's the algorithm used in the implementation of this thesis. The problems taking this or any other algorithm to use with the data at hand is that there is no direct way of verifying the effectiveness of it since there is no faulty data. So the only way to use the system at this point is to limit is as best as possible so that it doesn't characterize normal data as

anomalous, but is as sensitive to the future anomalies as possible.

3.6 Statistical models inside Greenhouse system

In order to be used, the Greenhouse system needs a few different statistical methods to be used in the process. These methods are being briefly presented in the next sections.

3.6.1 Multivariate normal distribution

Multivariate normal distribution is a widely used method in statistics. For example in situations where analysis of data is needed and can be thought as normally distributed. Multivariate normal distribution is the higher dimension version of normal distribution. (Tong 2012, p. 1-5)

$$\mu = E[X] = [E[X_1], E[X_2], ..., E[X_k]]^t. \tag{10}$$

Normal distribution is based on the mean and the covariance of a data. Mean of the data is calculated using the equation 10. There X is a k-dimensional vector and E is the mean function. (Bishop 2006, p. 19-20)

$$\Sigma = E[(X_i - \mu_i)(X_j - \mu_j)]. \tag{11}$$

Equation 11 has the covariance function. There Σ is the covariance. Covariance is kxk sized. (Bishop 2006, p. 19-20)

Normal distribution has no limits on the data so the values can run between $-\infty$ and ∞. That's why there's truncated normal distribution. In truncated normal distribution the distribution limits can be adjusted if needed to the level that is wanted. (Burkardt 2014)

3.6.2 Mahalanobis distances

Mahalanobis distance measures how many standard deviations away a point x is from a another point (Mahalanobis 1936). In this thesis the point x will be calculated against the mean μ. It can be used on a multi-dimensional level.

$$D_M = \sqrt{(\vec{x} - \vec{\mu})^T \Sigma^{-1} (\vec{x} - \vec{\mu})}. \tag{12}$$

Equation 12 has the M-distance where D_M is the M-distance, $\vec{x}$ is a point vector and $\vec{\mu}$ is the point vector of the mean. (Mahalanobis 1936)

3.6.3 Inverse cumulative distribution function

When a limit for a certain percentile is needed inverse cumulative distribution function finds that limit. It finds the closest point where given percentile of normally distributed data is included (Shaw 2006). For this it's a good function to use to find outliers of the upper or lower limits of normally distributed data.

4 IMPLEMENTATION OF MODEL

In this chapter analysis of data from breakers and relays will be done. A model for predicting anomalies in breakers will be implemented. This is being done according to the Greenhouse system selected in section 3.5. For this a LSTM prediction model is needed, and M-distance and multi-variate normal distribution has to be implemented on top of that to create the Greenhouse system.

4.1 Used environments

For the analysis Databricks environment will be used. Databricks is an Microsoft Azure cloud application. It has built-in many features used in data science. For example it has Apache Spark, Python, Dataframe and built-in. Also it is easy to access the data that is stored in Azure data storage. Clusters can be flexibly distributed for the data processing and it's really scalable in terms of computing power.

Since computing is being clustered, there is a need for a ML framework that can support the usage of many computing cores. Tensorflow and Keras are the most used ones (*Why use Keras* n.d.). Keras is a higher level deep learning framework that uses a simplified syntax compared to Tensorflow and uses Tensorflow as its main backend (*Why use Keras* n.d.). This is why Keras will be the ML framework used to create the model used in this thesis.

There are two cluster types that will be used in this thesis. The first one is the smallest cluster available on Azure, where one unit has 4 core CPU and 14 GB of ram. There will be 2-3 units on this cluster. This cluster is chosen as the minimum to test new code and to test the performance of different algorithms. The second cluster is the cluster that handles the harder workload like the learning process. Each cluster unit on that has 2 s and 112 GB of ram. There will be 2-3 in use for the cluster.

The programming language used in this thesis will be Python. Python is a widely used and ranks as third most used programming language based on the TIOBE-index (*TIOBE-index* n.d.). Also it supports the most popular ML frameworks. Other libraries used in this thesis includes Apache Spark for data access in cloud. Since Keras doesn't support the use of Spark Dataframe, Numpy will be used for data handling in the LSTM algorithm development.

4.2 Data preparation

Before the actual ML algorithm implementation can begin, a analysis and preparation on the data at hand has to be made. This is since the data is split up in different signals. Also it is appropriate to make the assumption that there is data missing here and there since the data collection can't be perfect at all times. This data ununiformity has to be addressed. In the Greenhouse system, LSTM model predicts two future values based on a time point predicting the value at the time point and one value after that. The model does the prediction based on three values that come before the selected time point as seen from the table 4.1. After prediction, an error vector is calculated based on the real value and the predicted value.

Table 4.1. LSTM inputs and outputs and the way error vectors are calculated in this thesis.

Previous values	Future values	Time point	Error vectors (t)
v_1, v_2, v_3	$p_4, p_{5.1}$	4	$v_4 - p_4, v_5 - p_{5.1}$
v_2, v_3, v_4	$p_{5.2}, p_{6.1}$	5	$v_5 - p_{5.2}, v_6 - p_{6.1}$
v_3, v_4, v_5	$p_{6.2}, p_{7.1}$	6	$v_6 - p_{6.2}, v_7 - p_{7.1}$
v_4, v_5, v_6	$p_{7.2}, p_{8.1}$	7	$v_7 - p_{7.2}, v_8 - p_{8.1}$

For data transformation Databricks uses spark framework for reading the data. This is really fast and can read 250 MB in under 2 seconds from the Azure Blob storage. This would then be really ideal for the use of this thesis. The drawback is that the usage is for time-series data and Spark DataFrame doesn't really have as much built in operations that can be used to morph the tables from signal level towards the needed format for the LSTM algorithm. For this reason the data read from the Blob storage is converted into a Numpy array. With the Numpy array in use, the data processing for one current signal takes around 5 minutes depending on the exact size of the signal file. The performance is not increased between different clusters in use so the performance is limited by the platform or the library itself.

In this thesis the signals will be used independently from each others. This is due to four facts. First the model needs to be tested in order for the maximum capability for prediction. Secondly it isn't known how much the signals help the other in the prediction and which of these signals are actually helpful to create the analysis. Thirdly some of the signals have so much data that in preliminary tests going through one epoch on one signal would take somewhere close to 17 minutes. Also it's not on the scope of this thesis. Validation for the LSTM model will be around one percent of the learning material. This will give a picture of how well it performs on data

that isn't being learned on without taking too much of the learning process.

4.3 LSTM model

After the creation of the input data for the system. A LSTM model is created. Like discussed in the section 4.1, Keras is used for the model creation. Performance is a really big factor in developing the LSTM model. That's why it's necessary to try out which parameters in the model affect the most on the performance of the training. Adjusting batch size, complexity of the model and number of epochs are being used. Batch size means the amount of input values that is being forward propagated before back propagation and weight update. In this case complexity of model means the amount of hidden layers in use and the amount of memory units in use. Epoch means the amount of times the that the training of the NN has gone through the learning data.

4.3.1 Limitations

In order to make the thesis more manageable to finish, some limitations are put in place. First, the devices for the LSTM learning process and the later normalization will be picked only based on the alphabetical order to simplify the choosing process. This might mean that the distributions of the different packs of data might be different. This also might be a possibility to see how much different environments affect the predictions.

Also only one signal is to be used in this thesis. After initial tests, the time needed for the processing of data is a lot. Also one signal can give a good picture already on the application of the algorithm and can be later easily applied to large variety of similar time series signals. Initial test for performance and usability have been done on the L2 current, travel time during closing/opening signals, number of CB operations and trip history. Other than the current value, signals don't seem to be working well with the chosen limitations. This is because of the characteristics of the data. Even though many of the signals are being collected every second, only the values that have change from previous values are being saved. In the cases of other than the current signal, there are rarely changes in the signal which creates a hard to predict environments since the frequency of the datapoints changes from time to time.

So even though other signals than current signal might be much better fit in order to catch the faults inside of breakers discussed in chapter 2.1.2, L2 current is the only signal that has consistent data needed for the algorithm chosen for this thesis. For this reason the analysis

will be based on only one signal that is the L2 current signal of the Emax 2 breaker. The L2 current signal describes the actual values of current that goes through the CB at any time. For actual CB fault detection the 1 Hz sampling is far too slow since the operation speed of breakers is maximum of tens of milliseconds (Callavik et al. 2012). In order to capture the opening and closing of the CB, there would be need for sampling rates in the realm of 10 kS/s. Still analysing the current signal at this rate might be helpful in finding problems and anomalies in a larger scale around the whole electrical system, its configuration and load. This might give the opportunity to catch problems happening in the whole electricity distribution network, but might also obfuscate the different faults when there are so many components in the network.

Since there would be too many variables to test through, if all variables would be tested through, some simplification are necessary. The amount of hidden layers will be limited to one and two for simplification and performance purposes. Also in initial testings the amount of memory units doesn't seem to limit the performance too much and more memory cells will be able to handle more complex information so after initial testings it is set to 512 memory cells. Also the input and output signal amount will be limited to the amount specified in the greenhouse system.

The worst part performance wise and also the most crucial parts for successful training a LSTM model are the batch size and the amount of epochs. Batch size should be small enough for efficient convergence and high enough so it wouldn't take too much time to process. The amount of epoch has to be adjusted so that it helps convergence though not too high since the number of epoch multiplies the time used to process the training data according to initial testing.

4.3.2 First tests

Initial tests are done with the Emax 2 current signals since those are the only raw values, where the others are calculated KPIs. The downside to using these values is that the amount of data is huge and running even one epoch of one breaker with one signal takes close to 10 minutes to get any results. Slicing down the amount of data might affect the representation of the larger scale so the optimization for the whole system.

First tests with larger amount of devices produces the fact that the loss of the training doesn't really go down from the initial loss and mostly hovers around. Also the accuracy stays quite low around 50 %. For the validation set the accuracy is a bit higher at around 51 % There might be few different reasons for this.

The optimizer might be wrong for the data used in the test or the settings used for it might be off and there might be too little signals to forecast future values from. Also the batch size might affect the learning process. These different possible causes will be tested in order to maximize the accuracy of the algorithm and help the fault detection algorithm to be as exact as possible.

The figure 4.1 have been plotted based on the error vectors of the current forecasts of the LSTM algorithms. The x-axis shows the error for the time t_1 and the y-axis shows the error for the time t_2. Raising the complexity of the LSTM algorithm is another way to try to better the forecast. First way to try to increase the complexity is by testing the train with larger amount of memory cells. Based on a train comparative to the first two tests doesn't lower the loss or raise the accuracy at all. Raising from 512 to 1024 memory cells only increases the time needed for training.

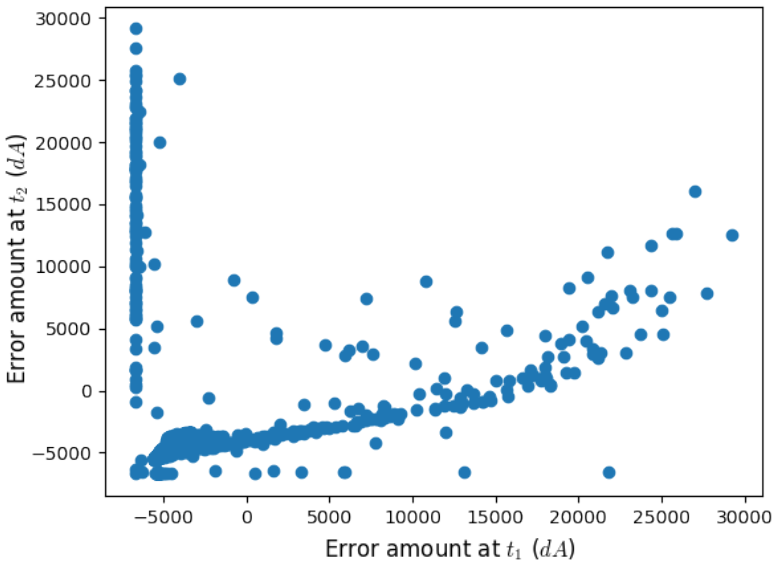

Figure 4.1. First test error scatter.

The second way to increase the complexity is by raising the amount of hidden layers. A test is done to add one other hidden layer to see how that affects the loss and the prediction rate. In order to keep the performance as high as possible, the amount of memory cells are cut down to 100 cells per hidden layer. Adding a second hidden unit seemingly works much better than having only one hidden layer since the loss keeps lowering a lot after even a few epochs surpassing one hidden level algorithm loss level many times over. The problem hits when trying to predict values. It slows down too much since predictions for one device's signal takes more than 30

minutes.

For a second try with two hidden layers memory cell count is cut down to 10 to increase the performance. With this the prediction performance decreases to around 9 minutes which is much more tolerable for application purposes. This test also shows that it might take considerable more time or resources to operate. More extensive tests on more complex model might yield better results, but won't be discussed more deeply in this thesis. In the test with two hidden layers in the algorithm, there is much more precise prediction. As seen from the figure 4.2 there is much more accurate prediction of values compared to the two test from before that are depicted in figure 4.1. There is two things to take out from this. There is strong correlation between the first and second predicted values based on the shape of the figure 4.2.

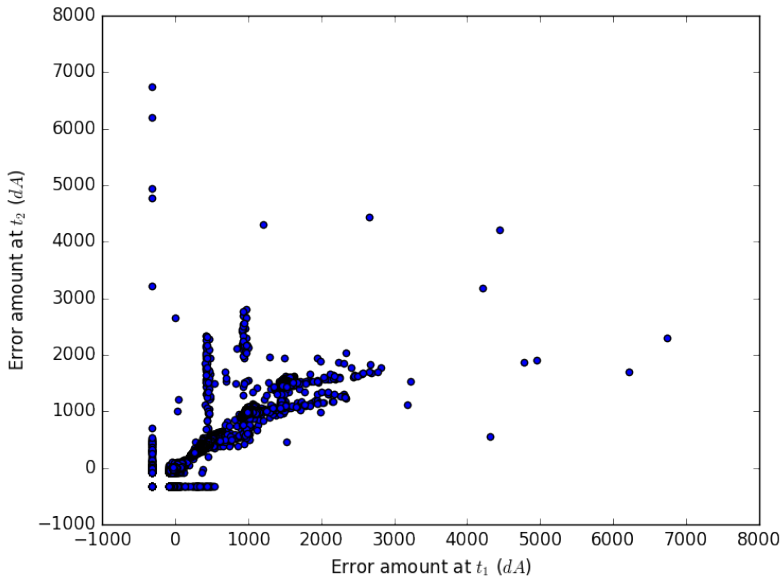

Figure 4.2. Error scatter with 2-layer LSTM algorithm.

Mostly also seems to be quite normally distributed. There are some straight lines which seem to point to a recurring pattern of some sort where for the other time point the prediction error is always the same in these points and for the other time point the prediction error changes somewhat places. It might be worthwhile to look into this if this pattern occurs on other signals as well. There are only a few predictions that are really off in terms of prediction error distribution. These errors will be looked into. If those far off errors are data handling related, it would better the prediction to take those into account in the data processing. One that situation where this kind of errors might be generated might be due to situations where the current signal

starts to rise from zero state at the time t or t_1 so the input time-points t_{-3}, t_{-2} and t_{-1} are still zero.

Evaluating the model with one of the largest breakers which has quite different distribution of data having one of the largest current values reveals that there is a big problem with the prediction and the real live signal. Looking at the table 4.2 it is plain to see that the model can't really seem to be able to predict values even roughly. This means that another approach is needed to achieve a good prediction.

Table 4.2. A few predictions of a large breaker.

Actual value t_1	Actual value t_2	Predicted value t_1	Predicted value t_2
6950	6826	1367.12	1367.1196
6826	6764	1367.12	1367.1196
6764	6781	1367.12	1367.1196
8183	8170	1367.1201	1367.1196

4.3.3 Final model

The most probable reason for the problem with the LSTM model seems to be the lack of signal pre-processing. As said in (Bishop 1995, p. 296), in many cases this is needed in order to help the algorithm to converge. It is hard for the model to process the current values because the values are so huge. For this reason, min max scaling is used between 0 and 1 for the data. This means that all the values will be scaled between 0 and 1 so that there is not large differences between values. When a prediction is done, the values are turned back to normal to be used in the later processes.

The LSTM settings are kept at the same level as previously since the same loss reduction amount most probably applies to also pre-processed data and a complex model is usually better learning capacity wise compared to a shallower one. The final model is being trained also on more varied data. This is better for distribution, but also limiting on the amount of data that can be used. All the different devices' L2 Current signal will be used to train the model. Since that is a lot of data, the data will be limited to 5 days of data that is used for training in order to limit the time needed for the training process.

Using the same test data that has been used in the first test on the final model, figure 4.3 is based on the errors of prediction. It looks much more evenly distributed and the clustering is much

more closely distributed towards zero. The data pre-processing seems to have a huge impact on the model's capability to do predictions.

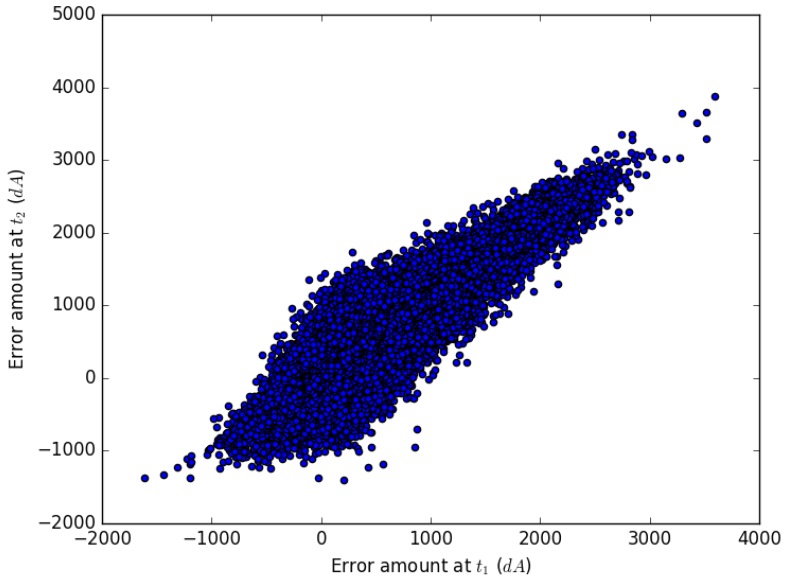

Figure 4.3. Error for a signal.

With the final model the predictions are more closer to the real counterparts though still not 100 % accurate as seen in the table 4.3. The scale of predictions are close and probably perfectly good at making rough estimation of the faults. Using much more data for prediction might yield better results and can be though for future implementation.

Table 4.3. A few predictions of a large breaker based on the final model.

Actual value t_1	Actual value t_2	Predicted value t_1	Predicted value t_2
5911	5897	5746	5630
5897	5868	5779	5660
6193	6175	6003	5881
6175	6145	6059	5931

4.4 Normalization and limit definition

In this section the other parts of the Greenhouse system other than the LSTM prediction model will be generated. To test out the multivariate normal distribution, mahalanobis distance and truncated normal with inverse cumulative distribution, signal of one device will be analyzed. Also the fully trained LSTM algorithm will be used. After this section a limit for faults will be defined.

4.4.1 Multi-variate normal distribution

The most important part of calculating the normal distribution is creating the covariance matrix. That is since it depends on the mean and the input data. On top of that the process of processing and predicting each signal takes a lot of time. That's why all the predictions will be saved as parquet files and processed later. That's why the predictions will have to be done only once and for the later processes Spark Dataframe can be used which is a lot faster than Numpy at data processing.

For the second part of the training dataset another part of the data is taken. Like in the LSTM model creation, a 5-day part of every device's L2 Current is taken to predict based on the model and then and then prediction errors to be calculated. Table 4.4 shows the mean and the covariance matrix of the data.

Table 4.4. The mean and the covariance matrix for the second training dataset's prediction error.

Mean (dA)	263.13	347.75
Covariance (dA)	339208	391179
	391179	483472

To get a sense of the normalization of the error signals first a contour of the error distribution will be constructed. In the figure 4.4 normal distribution is shown. The data shows that the normal distribution is quite highly correlated and the variance between prediction errors correlate quite well. This is a good thing for the analysis, since a coherent solution most likely can be found with normalization.

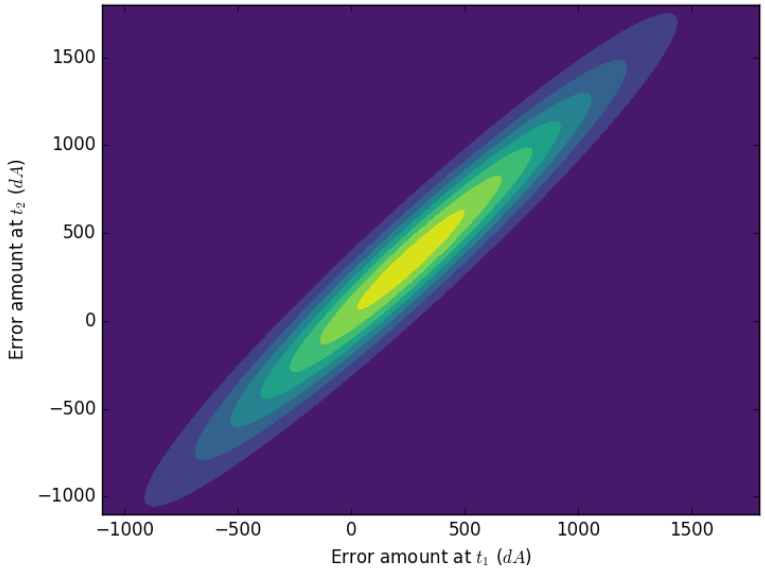

Figure 4.4. Normal distribution of error prediction.

What can be also found from the distribution is that the mean is not zero. That means that the predictions are a bit lower than the actual values. The mean isn't though too far since mean is 263 for t_1 and 347 for t_2. Also it can be seen from this mean that it is a bit harder to evaluate the t_2 prediction since the error is more than t_1. This follows the logic that further away future is harder to see than the nearer future.

4.4.2 Mahalanobis distances and inverse cumulative distribution

Next dataset 3 is taken similarly as the first two by taking the L2 current signal data from all devices from 5 day period that is different to the other datasets. From this data prediction error is calculated. From these errors mahalanobis distances are calculated based on the mean and covariance calculated with the second set of data and fitted to a inverse cumulative distribution function of a truncated normal distribution. Table 4.5 shows the results of the inverse cumulative distribution. The low end and upper end of the data handles the same way so same exclusion limit applies to both ends. The table shows that, if 99.99 % of data is deemed as the limit of faulty data, the mahalanobis distance on this limit is going to be maximum of 3.72. Based on this a limit for fault diagnostics is chosen.

Table 4.5. The mahalonobis distances for devices in order to belong into certain percentage.

Below (%)	0.01	0.1
Result	-3.72	-3.09
Above (%)	99.9	99.99
Result	3.09	3.72

4.5 Testing devices fault detection

Now that the limits for the model have been defined, fault detection on devices can be tested and adjusted more deeply. There will be two different signals that will be looked into more deeply. The first signal is from one of the largest breakers on the system which load is really stable so the data doesn't have much fast changes. The second signal being inspected in this thesis is from a device that is a much smaller than the first device. Also it's used in a process that has a lot of fast changing signal values.

What is the things that are being inspected in these cases are that how well the algorithm can predict these quite different signals and how much fast cycling data affects the prediction versus more stable data. These will most probably be seen from how large is the standard deviations and the means of the prediction error. If the mean is far off, but it's really closely clustered, then the model can't predict that size of signals that well, but does predict those signals reliably. If the mean is close to zero, but spread far and wide, it means that it predicts the size well, but doesn't do it that reliably.

There shouldn't be that much effect on the current levels of respectable signals since those are being scaled between 0-1 only based on that data alone. So the data on every device moves between 0 to the maximum value. The fault tripping will be compared to the dataset 2 based distribution, but to get an idea of the distributions of the test devices a distributions for both are done to get an idea of how it compares to the distribution of the dataset 2 in the figure 4.4. Drawing distributions for first signal 4.5 and for the second signal 4.6 shows that the distributions are less spread than the distribution based on all the devices. The signal two distribution though seems to be more wide and spread more apart from the correlation axis which is $x = y$. This might be because of the more cyclic nature.

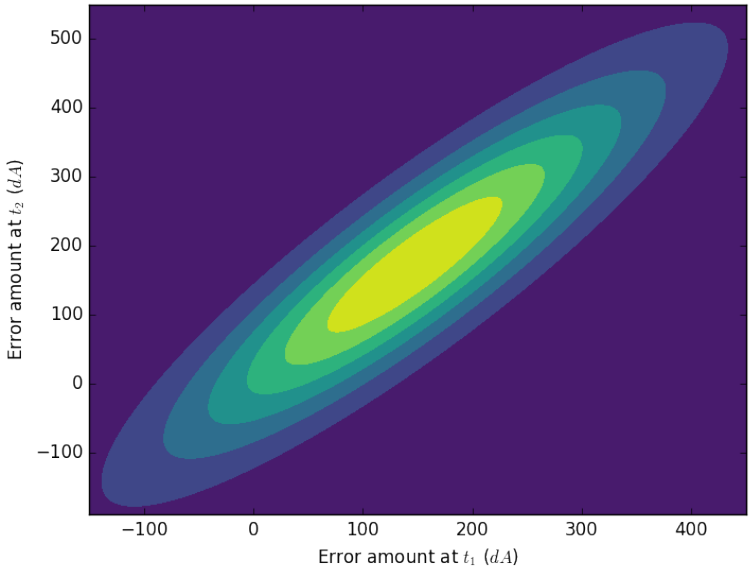

Figure 4.5. Signal one prediction error distribution.

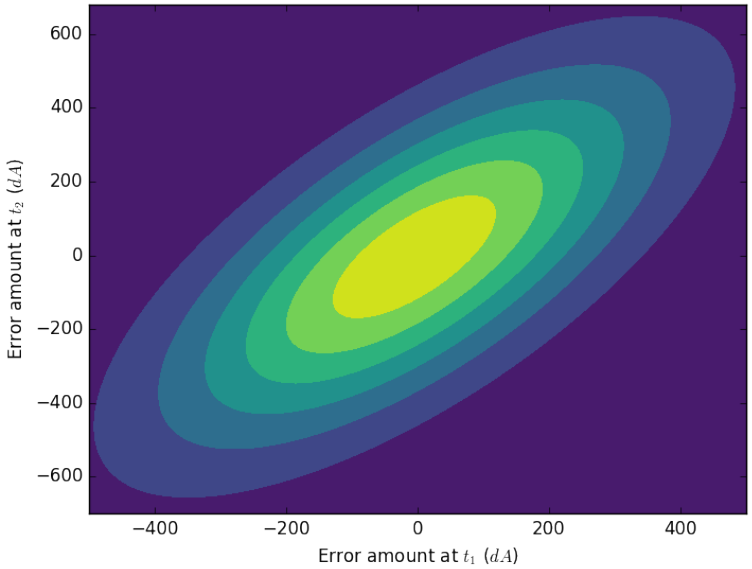

Figure 4.6. Signal two prediction error distribution.

In terms of making better predictions. For both of these signals the first fix to would be to expand the model to include far more training data. At this point the model is only based on 5 days worth of data from a bit over 100 devices. It's already huge amount of data, but only a fraction of the collected data. For that the prediction is quite good already.

4.5.1 Fault generation for test signals

The next part of the fault prediction is to calculate the M-distance for all data points and compare those distances to the limit that has been set previously based on the training datasets. The amount of fault trips should be zero since there haven't been any problems in the devices in question. After the fault testing some faults are being analysed and reasons and possible ways to combat possible problems will be discussed. This is where the possible problems would come out if there would be any. With the set limit there are 953 trips for the first tested signal. For the second signal there are around 142000 trips. So based on this, the cyclic signal from test device 2 is far less well predicted than from the more stable test device 1.

In the first device there are some fault trips that happen because the device data collection has stopped and part of the prediction data is from before the collection stop and part after it. In order to minimize the effect of these drops of data collection, all the predictions affected by the gaps will be removed from prediction. This is done by deleting all the dataset data points that has more that have more than 5 seconds between the first t_{-3} and the last t_1 data point. For normal data collection the distance should be 4 seconds, so the 5 seconds gives 1 second leeway, which shouldn't much affect the prediction.

After running all the steps for signal two with the data what happens is that the amount of fault trippings raise after the change. Looking more deeply in the data there is one obvious reason. Checking data points with high M-distance there is the same pattern happening all the time. One example is on the table 4.6. Here data collection stops abruptly and for that reason prediction goes far off. This is since LSTM uses memory of previous values to affect the prediction of each values.

Table 4.6. Signal collection break.

Time (hh:mm:ss)	Signal value (dA)	Mahalanobis distance
06:05:31	1434	0.279
06:05:32	1433	0.288
06:05:33	1435	0.275
06:05:34	0	7.163
06:39:49	9366	skipped
06:39:50	24091	skipped
06:39:51	25778	skipped
06:39:52	25346	87.9
06:39:53	24628	26.1
06:39:54	23799	24.7

In this case the values that come before the stop still affect the prediction so the prediction is really off at the beginning and then starts to level off. In this case it takes 10 predictions until it levels off after the abrupt stop. Also it seems that the last signal before the stop with the signal value 0 means either that the breaker current is stopped abruptly or then it might be a faulty signal.

In itself the signal seems to be working fine. It raises over the trip limit when there is a big difference in data, but connection problem easily hides the device related problems since there seems to be a lot more of those compared to possible other fault signals. This is why these data collection related problems should be taken or at least mark them so a deeper look of the actual prediction can be made. For now these data collection related faults are being hidden.

By hiding signals with high M-distance distance after a 30 second or more break in data collection, limits the amount of fault trips to 139000 trips so the effect is noticeable, but doesn't change much. This isn't in itself a huge number being 0.75 % of all data points, but in still far too much faults on a healthy system. On the first signal these changes lower the amount of trips to 909 trips which is 0.0039 %.

of the whole data of the device. There's not a lot of change, but still noticeable and the fault amount is already really good and shows that the algorithm is able to learn the signals quite well at least for a more stable system.

Next thing is to figure out is to see what other kinds of failures does the algorithm produce. For the first signal many of the fault signals happen when the current suddenly drops or raises the

current value magnitude of around 100 A based on a few trips analysed. The same happens when analysing the second device. Analyzing few devices shows that the limit is passed when the signal raises or lowers close to 100 A or more between signals. This is quite big current change inside one second though it is characteristic to the second test device. In many of the fault trips the change has been even more than 33 % on the second device.

All in all the algorithm seems to work as intended. It raises vastly different signals and signal changes to light. Changing the limits might give a better or worse picture of the situation of the signal. For example, if the second test signal M-distance limit is set to 10, there is only 8370 faults found. Also the faults might be a lot more limited, if more training data is used and more of the cyclic type of signal data is added to the training set.

Also these trips that are based on big drops or hikes in signal values beg the question that is 1 second sample rate enough for the current signal to get an accurate picture of the characteristics of the data. Electricity reacts quickly so the signal might also need to inspect a lot faster. This algorithm framework is really easily applied to other signals that have similar type of data. Also dividing devices that have different characteristics might make the prediction better and make it even easier to find problems in those types of situations. Even an algorithm based on one signal itself might provide good results and might worth to investigate.

5 DISCUSSION

The algorithm implementation done in this thesis was with its limitations successfully done. It has the ability to detect outliers from signal data. The algorithm doesn't though really answer are the circuit breakers faulty or just having abnormal, but acceptable behaviour. Also having chosen the current as the signal under observation, the faults are more likely to come also from other components in the electricity distribution. The algorithm can't straightaway answer the question might there be one of the most probable faults happening in the devices. This is a limitation and one of the future directions that should be researched to improve it to the direction where the algorithm would itself point to a direction where the fault might be originating from. Because of the limitations, a direct analysis couldn't be made from the device faults and that needs to be done further.

Also by improving the prediction might give information about actual faulty signals and this might be implemented later to see the devices that might break down. This would be the ultimate goal for the customer to get a list of devices that have to be maintained or replaced in order to be run for the next two years. There might be limitation on the signal level also. The algorithm can't use data past the sampling rate so, if there are changes that happen faster than the sampling rate, those are left out of the analysis. Also because of this the real behaviour might obfuscate and hide the real faulty signals.

As of now the implemented algorithm only includes one signal. It would be might be benefactory to test how much better predictions could be attained by including more signals into a single model since those signals might reveal outlier values better than bunch of models based on single signal per model. For the future online data handling might give an added benefit. This might give added value to the customer when a algorithm can run real time or almost real time and give alarms if needed so quick actions can be done if necessary.

Also it might be benefactory to research, if it would better the fault detection, when the different styles of signals would be divided into different algorithms or even all, if a model could be done for each device. This would make it easier to concentrate really on the specific parts of the data, but might also easily make it too prone for fault tripping.

This implementation of the algorithm was way too slow in terms of data processing since

converting data from Spark DataFrame to either Numpy or Pandas DataFrame was really slow. It ended up taking usually half of the time of the model creation and most of the time while doing predictions. This couldn't be skipped since Keras doesn't work with Spark DataFrame and also the Spark DataFrame library doesn't include a lot of needed functions that make time series data handling much easier.

A lot of manual work was needed for the parts of data processing that used Spark DataFrame. Finding a machine learning library that supports Spark DataFrame or using a platform that enables file reading with other than Spark DataFrame could make the data handling much better. In the model created in this thesis the multi-threading was somehow not fully working since the learning speed of the model wasn't hugely affected by the size of the cluster. For further development this should be fixed so the model creation speed could be raised.

6 CONCLUSIONS

The aim of this thesis has been to open the road towards the world of predictive maintenance by learning about the devices for the thesis, describing and analysing different styles of machine learning algorithms, choosing the right ML algorithm based on the limitations and implementing the chosen model. The devices that have been analysed for this thesis are the circuit breakers that are ABB produced breakers with model name Emax 2. The main problems with circuit breakers has been defined and the problems mainly concentrate on the mechanical switching device.

Choosing the ML algorithm has been done by introducing different fields of ML and then narrowing the best directions to choose algorithm from based on previous studies around the subject. SML was better style of ML since there possibility to automate these processes far better and also it gives the ability to develop the algorithms to characterize the findings much better. This can give the ability in the future for a fully automated system which needs as little human interaction and thus can be more scalable.

With the limitation of having only data from a new perfectly working devices, an algorithm that can be formed based on good data was needed. This is why Greenhouse system was chosen as the best algorithmic solution. It is a LSTM based solution that uses the LSTM algorithm to predict signal values. The predictions are compared to the actual values and an error is calculated. This error is normalized and then signal values that have high standard deviation are being flagged. This also gives the ability to use SML algorithm on unlabeled data.

In this thesis an algorithm based on the Greenhouse system has been created. This has been done by analysing a single signal being a current signal from a rather large number of similar devices from a pulp mill. This current signal collects values of the day to day current load on the CB. The dataset includes devices from different processes having quite different characteristics in the operational behavior of the load of the devices. For model creation three datasets based on five day sample of all the different devices in the data pool at hand was used.

The operation has been tested on two completely different, but working devices. One with high, but stable current signal and a second that has low, but cyclic current signal. The more stable signal proves to be significantly easier signal to predict than the cyclic signal. With the algorithm

created that has been created for this thesis the first test signal trips 0.0039 % of the data and the second test signal trips 0.75 %.

The algorithm then really has learned normal usage for the current signals. The development isn't though ready for any kind of real application as of yet, since it trips from the normal usage far too often still. It is though a good start for a framework that can be applied to many different value based signal collection, if applied well and developed further. As of now there are still too many trips over the limit especially on the second signal that has been tested so that should be the focus, if future development is being done on the algorithm.

At this level of implementation the algorithm is too simple to purely work on its own. It still needs a specialist with substance knowledge to interpret the fault trips. Also due to the limitation of only using the current signal, it has has been hard to find the mechanical problems of the CBs so the algorithm mostly sees problems on a systemic level. This means that the algorithm marks anomalies that might be caused by any device that has an affect on the current.

REFERENCES

ABB (n.d.). *SACE Emax 2*. URL: https://new.abb.com/low-voltage/products/circuit-breakers/emax2.

AI Impacts (2015). *Recent trend in the cost of computing*. URL: https://aiimpacts.org/trends-in-the-cost-of-computing/.

Amazon (n.d.). *AWS Pricing*. URL: https://aws.amazon.com/pricing/.

Bishop, Christopher M. (1995). *Neural networks for pattern recognition*. Oxford: Clarendon Press.

— (2006). *Pattern Recognition and Machine Learning*. 1st ed. 2006. Corr. 2nd printing. Information science and statistics. Springer.

Bontemps, Loic et al. (2017). "Collective Anomaly Detection based on Long Short Term Memory Recurrent Neural Network". In: *CoRR*. URL: http://arxiv.org/abs/1703.09752.

Brown, Richard E. (2002). *Electric Power Distribution Reliability*. Power engineering. Marcel Dekker.

Burkardt, John (2014). "The truncated normal distribution". In: *Department of Scientific Computing Website, Florida State University*.

Callavik, Magnus et al. (2012). "The hybrid HVDC breaker". In: *ABB Grid Systems Technical Paper* 361, pp. 143–152.

Chandola, Varun (2009). "Anomaly detection: A survey". eng. In: *ACM Computing Surveys (CSUR)* 41.3, pp. 1–58. URL: http://cucis.ece.northwestern.edu/projects/DMS/publications/AnomalyDetection.pdf.

Cloud, IBM Marketing (2017). *key marketing trends for 2017 and ideas for exceeding customer expectations*.

Connor, J. (1991). "Recurrent neural networks and time series prediction." eng. In:

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. http://www.deeplearningbook.org. MIT Press.

Hochreiter Sepp; Schmidhuber, Jürgen (1997). "Long Short-Term Memory". In: *Neural Computation* 9 (8). DOI: 10.1162/neco.1997.9.8.1735.

Janssen, Anton (2014). "International Surveys on Circuit-Breaker Reliability Data for Substation and System Studies". eng. In: *Power Delivery, IEEE Transactions on* 29.2, pp. 808–814.

Kingma, Diederik P. (2014). “Adam: A Method for Stochastic Optimization”. In: URL: https://arxiv.org/abs/1412.6980v8.

Kubat, Miroslav (2015). *An introduction to machine learning*. Springer.

Lee, Tae Jun (2018). “Greenhouse: A Zero-Positive Machine Learning System for Time-Series Anomaly Detection”. In:

Leslie Hewitson Mark Brown, Ramesh Balakrishnan (2005). *Practical power systems protection*. Practical Professional Books. Newnes.

Lindquist, T. M. (2008). “Circuit breaker failure data and reliability modelling”. In: *IET Generation, Transmission and Distribution* 2.6, pp. 813–820.

Mahalanobis, Prasanta Chandra (1936). “On the generalized distance in statistics”. In: *Proceedings of the National Institute of Sciences (Calcutta)* 2, pp. 49–55.

Malhotra, Pankaj et al. (2015). “Long short term memory networks for anomaly detection in time series”. In: *Proceedings*. Presses universitaires de Louvain, p. 89.

Nilsson, N (1998). *Introduction to Machine Learning*.

Olah, Christopher (2015). *Understanding LSTM Networks*. URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Rojas, Raúl (1996). *Neural Networks: A Systematic Introduction*. eng. URL: https://wilma.finna.fi/lut/PrimoRecord/pci.springer_s10_1007_978_3_642_61068_4.

Shaw, William T (2006). “Sampling Student’s T distribution-use of the inverse cumulative distribution function”. In: *Journal of Computational Finance* 9.4, p. 37.

TIOBE-index (n.d.). URL: https://www.tiobe.com/tiobe-index/.

Tong, Yung Liang (2012). *The multivariate normal distribution*. Springer Science & Business Media.

Why use Keras (n.d.). URL: https://keras.io/why-use-keras/.

Zazo, Ruben et al. (2016). “Language Identification in Short Utterances Using Long Short-Term Memory (LSTM) Recurrent Neural Networks”. English. In: *PLoS One* 11.1.