NTNU – Trondheim
Norwegian University of
Science and Technology

In-depth Exploration of Machine Learning In Predictive Maintenance

Kolbjørn Trøstheim Flaarønning

Autumn 2019

TPK4530 - Specialization Project
Department of Mechanical and Industrial Engineering
Norwegian University of Science and Technology

Supervisor: Jørn Vatn

RAMS
Reliability, Availability,
Maintainability, and Safety

Preface

This paper displays the work conducted as a part of the TPK4530 - Production Management specialization project. It was written in fall 2019 and functions as a prestudy to the Master's thesis at the Engineering and ICT study program in spring 2020. The specialization project is completed in collaboration with Teekay Shipping NYSE to combine the study of data-driven prognostics models and real life maintenance data. It's assumed that the reader has fundamental knowledge in the field of maintenance. Machine learning knowledge is preferred, though the most relevant information is provided in the paper, otherwise the reader is referred to other machine learning studies and theory.

Trondheim, 2019-12-18

Kolbjørn Trøstheim Flaarønning

Acknowledgment

This report was written in collaboration with the RAMS institute at NTNU, represented by my supervisor Jørn Vatn. I am grateful for the valuable insight from Jørn regarding report structure and content. Gratitude's goes to the representatives from Teekay for providing relevant data and discussion of potential Master's thesis objectives. Finally, I would like to thank my colleagues Mari Tuhus and Ellinor Wikan for their close collaboration and contribution to this report.

K.T.F

Executive Summary

Condition monitoring has become a vital maintenance strategy across many industries. Obtaining information regarding health and condition of system's components can heavily reduce the maintenance costs as well as reduce the risk of catastrophic failures. With today's emerging technology and particularly Internet of Things (IoT), it is possible to install sensors on every imaginable system component. This leads to better and more insightful system information, though it requires complex strategies for data acquisition and processing, as well as advanced predictive techniques.

This thesis explores the possibilities of using machine learning in condition-based monitoring. This involves diagnosing system's health and current condition, as well as predicting system and/or component failure. Continuous monitoring systems result in large and complex data, where traditional maintenance strategies might not be efficient enough. Machine learning strategies and especially neural networks have proven to perform well on complex data, with their ability to derive advanced patterns.

The first and main part of this thesis is a literature review concerning the use of machine learning in maintenance. The most developed applications are reviewed and show promising results. Especially the applications with implemented neural networks have a high predictive accuracy with respect to the prediction of remaining useful life of systems. However there are several questions that remain unanswered. The most common critique of machine learning models are their "black-box" property. Although the final prediction accuracy is high, its not easily tractable. The models don't explain how it derived that particular conclusion, neither how confident they are in their decision. Some of these challenges have to some extent been resolved with the implementation of probabilistic neural networks and uncertainty quantification.

The second part of the thesis is a case study where multiple recurrent neural networks are created to solve the 2008 challenge provided by the Prognostics and Health Management (PHM) society. The challenge involves predicting the remaining useful life of aircraft engines. Several preprocessing steps and recurrent neural networks have been implemented, where a Gated-Recurrent-Unit network had the best results. In addition, the confidence of the model and it's uncertainty have been assessed by using a Monte Carlo droupout technique.

Abbreviations

ADCC	Anomaly Detection With Changing Cluster Centers
ANN	Artificial Neural Networks
AI	Artificial Intelligence
ARIMA	Auto Regressive Integrated Moving Average
BP	Backward Propagation
BM	Breakdown Maintenance
DP	Damage Prognosis
DT	Decision Trees
DL	Deep Learning
EN-ELM	Ensemble-Based ELM
ELM	Extreme Machine Learning
FFNN	Feed Forward Neural Network
GRU	Gated Recurrent Unit
IOT	Internet of Things
KNN	K-Nearest Neighbours
LSTM	Long Short-Term Memory
MAP	Maximum A Posteriori
ML	Machine Learning
MSE	Mean Squared Error
MCD	Monte Carlo Drop-Out
MIL	Multiple Instance Learning
NB	Naives Bayes
NN	Neural Networks
PCA	Principle Component Analysis
PHM	Prognostics and Health Management
PdM	Predictive Maintenance
PM	Preventive Maintenance
RF	Random Forest
RNN	Recurrent Neural Network
RL	Reinforcement Learning
RUL	Remaining Useful Time
RMS	Round-Mean Squared
SBI	Similarity-based Interpolation
SLFN	Single-Hidden Layer Feed-Forward Network
SHM	Structural Health Monitoring
SRNN	Simple Recurrent Neural Network
SSD	Sum of Squared Difference
SL	Supervised Learning
SVM	Support Vector Machines
SVR	Support Vector Regression
USL	Unsupervised Learning

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

Maintenance is a critical aspect with respect to the reliability and availability of a system. As systems grow more complex their components become more interdependent, which results in a larger need of maintenance (Saranga and Knezevic 2001). With increasing global competition, organizations are forced to continuously enhance their maintenance performance, to reduce costs and improve productivity. Saranga and Knezevic (2001) refers to an increasing trend in cost of unplanned downtime and reveals that investing in condition monitoring and other maintenance management techniques can save the company a significant amount of money.

Traditional maintenance strategies like preventive maintenance or even unplanned maintenance has become too expensive for industrial companies due to the rapid change of technology (Martin 1994). It is therefore required to incorporate some sort of condition-based maintenance (CBM) strategy (Jardine, Lin, and Banjevic 2006). A CBM is a two-step maintenance strategy with system diagnostics and prognostics. System diagnostics involves acquiring data about the system's condition, in example detecting abnormal behaviour or identifying faulty components. Today's technology allows the implementation of sensors on every imaginable system component, which results in a complexity-level too great for traditional maintenance models (Accorsi et al. 2017). However, the increase in data ensures the opportunity for data-driven models to perform.

Machine learning is vastly used in anomaly— and pattern—recognition. These applications can be used in predictive maintenance by detecting failure patterns and provide early warnings. Though creating reliable and precise prediction models is not straightforward. The challenges, benefits and opportunities with these data-driven models are being investigated in this thesis.

.

1.2 Problem Formulation

This report functions as a prestudy for the Master's thesis, concerning the use of machine learning methods in predictive maintenance. Therefore, the objective of this report is mainly researching the use and the possibilities of machine learning within predictive maintenance. A comprehensive overview of deployed applications with different machine learning models are required. This involves examining the challenges and benefits of different models. Further, investigate the limitations of data-driven models and review what remains to be done.

1.3 Objectives

The main objectives of this project thesis are:

1. Review related work regarding the use of machine learning methods in predictive maintenance.
2. Identify the benefits and challenges for multiple machine learning methods
3. Identify the state-of-the-art applications of data-driven models in predictive maintenance.
4. Review the literature to find the limitations of machine learning in predictive maintenance.
5. Create and evaluate different machine learning models on relevant data.

1.4 Contributions

The contributions of this project thesis are:

- A thorough literature review explaining the most developed applications of data-driven prognostic models to date. Their benefits and shortcomings are compared and evaluated.
- A case study demonstrating how machine learning can be used in assessing the health status of industrial components and predict their remaining useful lifetime.
- Demonstration of probabilistic neural networks in order to evaluate the uncertainty of the model predictions.

1.5 Scope & Limitations

The scope of this project thesis is to investigate the use of machine learning approaches in predictive maintenance. This involves reviewing the most developed data-driven prognostic models. Which machine learning algorithms are most commonly used, what are their benefits and their challenges. Finally, identify the challenges of using machine learning in predictive maintenance and what needs to be improved to get a robust prediction model.

Reviewing the literature is not enough, it's strategies and decisions need to be tested. The second part of this report is a case study which implements the strategies from the literature. The end goal of the master thesis is to create a data-driven prognostic model on data presented by our collaboration partner Teekay. This paper displays some information on their relevant data, however a comprehensive data processing and filtering step is required. Hence, the case study is limited to other simulated data which acts as a test study to get insight in potential challenges. This experience will be valuable when implementing the final prognostic model on the Teekay data the following semester.

1.6 Outline

Chapter 1 is an introduction to the topic to be discussed in this project thesis, consisting of the background, main objectives to be solved, as well as the approach and limitations related to the work. Further, Chapter 2 introduce a general framework to relevant theory linked to the topic. In Chapter 3 an introduction to the previous work and literature connected to the work is reviewed. Chapter 4 is a case study which contains the case description, data evaluation and preprocessing techniques, choice of models and their respective architecture. The case study results are depicted in chapter 5. Chapter 6 discusses the findings in the literature review, as well as the result from chapter 5. Chapter 7 introduces our collaboration partner Teekay. Their expectations to the master thesis is discussed, as well as some insight into their data systems. Finally chapter 8 concludes the paper, summarizing how the problem objectives were met.

Chapter 2, 3, and 7 is written in collaboration with my student colleagues Mari Tuhus and Ellinor Wikan.

Chapter 2

Theoretical Framework

2.1 Maintenance

The evolution of maintenance strategies can, according to Yang and Widodo 2010, be summarized from breakdown maintenance (BM), preventive maintenance (PM) to predictive maintenance (PdM), where breakdown maintenance is also known as corrective maintenance or reactive maintenance. The ladder type of maintenance is essentially run-till-it-breaks, and the equipment is repaired or replaced only when obvious problems occur. Preventive maintenance, on the other hand, is known as time-based maintenance. The reasons for such maintenance are to prevent failure, to detect onset failure or to discover hidden failures. PdM, is also known as condition-based maintenance, which involves monitoring the condition and operation of equipment to predict whether it will fail during some future period, and take action to avoid the consequences of failure. In order to get a clear overview of the different stages of maintenance a definition of a few concepts is necessary:

- **Symptom:** An abnormal deviation of an observable quantity
- **Fault:** An abnormal deviation of at least one characteristics of a component or a system
- **Failure:** An event that puts a component or a system into a state so it permanently unable to perform its functions

The definitions were given in the course *TPK4450* by Anne Barros (Barros 2019).

2.1.1 Diagnostics

In order to understand the concept of diagnostics it's crucial to have a clear understanding of the three following definitions:

- **Fault detection:** To determine if a fault is present or not
- **Fault isolation:** Identifies the nature of the fault
- **Fault estimation:** An estimate of the magnitude or the severity of the fault

Hence, the definition of fault diagnosis includes all of the three previous given definitions, fault detection, fault isolation and fault estimation. Fault diagnosis is detecting, isolating and identifying incipient failure condition, while the affected component is still operational, even though at a degraded mode. Detecting a fault is to determine whether a fault is present or not, based on the observation of symptoms, whereas isolating a failure is to identify the fault component, sometimes referred to as fault location and fault estimation to estimate the magnitude or severity of the fault.

2.1.2 Prognostics

In maintenance one usually talk about failure prognosis, meaning the prediction of the remaining useful life of a failing component, system or sub-system. Prognostics focuses on predicting the time at which a system or a component will no longer be able to perform its intended function and is per definition an estimation of the remaining time to failure (Yang and Widodo 2010). Fault diagnosis is about implementing the given recommendations for when to perform the maintenance and to which extent this needs to be done, and failure prognosis is mainly about optimising the given recommendations.

2.1.2.1 Remaining Useful Life

The remaining useful lifetime (RUL) is defined as "remaining useful life of the system at time t given all available information up to time t" (Barros 2019). RUL is a random variable corresponding to remaining operational time left on an asset, and when taken into account engineers can plan maintenance and optimize operating efficiency. Relevant inputs in a RUL prediction are extracted features from sensors and condition indicators. The determination of RUL changes slightly depending on how the problem is addressed, either as model-based— or data-driven— prognosis, and what kind of data that is available:

- Lifetime data, contains data indicating the time to failure for similar assets.
- Run-to-Failure data, contains data of recorded assets to failure.

- Threshold data, contains predefined threshold values of condition indicators.

2.1.2.2 Forms of prognostics

In general, the aim in prognostics is to give reliable predictions about the RUL of a component or a system experiencing deviation or degradation from it's normal operating condition and intended function (Baraldi et al. 2013). In diagnosis the area of interest is connected to the question *"what is the current state of the unit?"*. Instead, prognostics focuses on answering the question *"what is the remaining time before the unit is in a faulty state?"* (Barros 2019). It is expected to improve planning of maintenance actions, increase safety and lower costs.

For the assessment of the development of failure of a degrading system, many different variations of data and information might be available. Based on the applied situation, prognostics can be addressed with different points of view and different prognostic methods my be applied. In order to classify prognostics methods they can be separated into two categories: *Data-Driven Prognostics* and *Model-Based Prognostics*. Data-Driven Prognostics methods deal with the transformation of the data provided by the sensors into reliable models that capture the behaviour of the degradation" (Tobon-Mejia et al. 2012, p. 492). By the use of machine learning techniques and pattern recognition, data-driven prognostics detect changes in system states. To predict future development of the degradation in model-based prognostics, explicit mathematical models are applied (Baraldi et al. 2013). In this way, it can incorporate a physical understanding of the system into the estimation of RUL. The aim is to use the dataset to fit a model for the system or component from the running state to the failure (Barros 2019).

2.1.3 Predictive Maintenance

Generally, the predictive maintenance framework is the combination of two key parts: prediction of the system remaining useful life time (RUL) and making decisions based on these predictions. Hence, the models used for creating such prognostic models are generally classified into two groups: model-based and data-driven models. Model-based approaches are stochastic models describing the evolution of system degradation in the discrete or continuous time. Data-driven PdM models use data to predict the system or components RUL without knowing the physical nature of the degradation mechanism (Nguyen and Medjaher 2019). Data-driven approaches for fault detection and PdM are usually based on data analysis and statistical techniques like machine learning algorithms. Such models are built using enormous amounts of historical data, a result of Internet of Things (IoT), referring to the increasing presence of things or objects, such as senors, actuators and mobile phones interacting with each other and hence generating vast amounts of data. Such large data sets are known under the term of Big Data, which may be analyzed to reveal patterns, trends and correlations. Data-driven methods for

PdM has become very efficient due to the improvement of artificial intelligence. Hence, industries and manufacturing companies are turning to Industry 4.0, also known as the Industrial IoT, with the aim to improve their maintenance strategies.

2.2 Uncertainty & Decision-Making

In prognostics one deal with prediction of the future behaviour of components, sub-systems or systems, hence there are also several sources that will to some extent influence the uncertainty of the prediction. Therefore it is useless to given an estimation of RUL without computing the uncertainty associated with it (Sankararaman and Goebel 2013). In order to make safe predictions one have to be able to formulate the uncertainty of the RUL prediction, either given by the density of probability or by the confidence interval. Some sources that need to be considered when it comes to the uncertainty of the prediction could include some of the following factors:

- Current State of the system or component
- Future Solicitation
- Degradation Process
- Concurrent faults
- Environment variations
- Sensor noise
- Measurement errors

Therefore, it is almost impossible to give a precise prediction of the RUL. Hence, one need to analyze how the previously mentioned sources might influence and affect the RUL prediction, and compute the overall uncertainty. Uncertainty quantification deals with identifying and characterizing the sources that might affect the RUL prediction, which should be incorporated into the model (Sankararaman and Goebel 2013).

2.2.1 Aleatoric uncertainty

Aleatoric uncertainty refers to the natural randomness of a system (Kendall and Gal 2017). If the same experiment, with same conditions and parameters are run multiple times, the outcome won't be exactly the same each time. Hence, the aleatoric uncertainty describes the uncertainty of observations y, due to a noisy data set (X, Y). The randomness given discrete variables, is given by the probability of each possible value. The randomness of continuous variables is parameterized by the probability density function.

It can be further split into heteroscedastic and homoscedastic aleatoric uncertainty (Kendall and Gal 2017). The heteroscedastic aleatoric uncertainty refers to a different level of noise for each observation (x, y), which means that the uncertainty of each observation is dependent on x. On the other hand, homoscedastic aleatoric uncertainty assumes that the uncertainty of each observation is the same.

2.2.2 Epistemic uncertainty

A model's epistemic uncertainty is due to limited data and knowledge (Kendall and Gal 2017). Causes of a models epistemic uncertainty can be a result of neglecting relevant aspects of the problem or inaccurate measurements. A typical example is the estimation of gravitational acceleration on the surface of the earth, where the effects of air resistance might be neglected to simplify the experiment. This neglection then results in a higher epistemic uncertainty of the estimation.

2.2.3 Modeling uncertainty in neural networks

2.2.3.1 Epistemic uncertainty in Bayesian Deep Learning

The epistemic uncertainty of a neural network can be modelled by placing a prior distribution over the model weights, and then observe how much these weights vary given input data (Kendall and Gal 2017). This distribution can be a Gaussian prior distribution. The model's weight parameters are no longer deterministic, but represented as a distribution over the weights, and the model is referred to as a Bayesian neural network. During training, the weights are no longer optimized directly, but rather averaged over the all possible weights. Given a dataset $x = (x_1, ...x_n)$, the model weights, $w = (w_1, ..., w_n)$ and corresponding labels, $y = (y_1, ..., y_n)$, the posterior distribution of the weights, $p(w|x, y)$ can be obtained by using Bayesian inference. This inference is calculated by using a dropout technique in the network. A dropout layer randomly deactivate a percentage of the nodes in the hidden layers (chapter 3.4). The dropout layers are used when training the model, as well as during the prediction of new samples, referred to as stochastic forward passes, or Monte Carlo droupout (Kendall and Gal 2017). Epistemic uncertainty can be reduced with the increase of data to the model.

2.2.3.2 Aleatoric uncertainty in Bayesian Deep Learning

As explained above, the epistemic uncertainty is a property of the model, while the aleatoric uncertainty is a property of the data. To obtain the aleatoric uncertainty the Maximum a Posteriori

(MAP) is used, where the goal is to find a single value of the model parameters (Kendall and Gal 2017).

2.3 Machine Learning and Its Applications

2.3.1 Machine Learning

Recently, it has been an increase in the use of machine learning (ML), a branch of AI, that can find valuable connections and insights from large sets of data. The increasing amount of data available, facilitated by the growing capabilities of hardware, cloud-based solutions and newly introduced state-of-the-art algorithms, is changing the industry and the way decisions considering scheduling, maintenance management and quality improvement is made (Susto et al. 2014). These data contains a variety of formats, quality, amount, semantics e.g. sensor data from a production process, environmental data or machine tool parameters. (Wuest et al. 2016). For instance, the availability of quality-related data offers potential to improve process and product quality sustainability.

By improving automatically through experience and learning from the surrounding environments, ML can design computer algorithms to emulate human intelligence, without being explicitly programmed (El Naqa and Murphy 2015; Mitchell 1997). The learning process starts by observing big data looking for patterns and connections, in order to obtain results from these patterns using various algorithms which can lead to better decisions in the future.

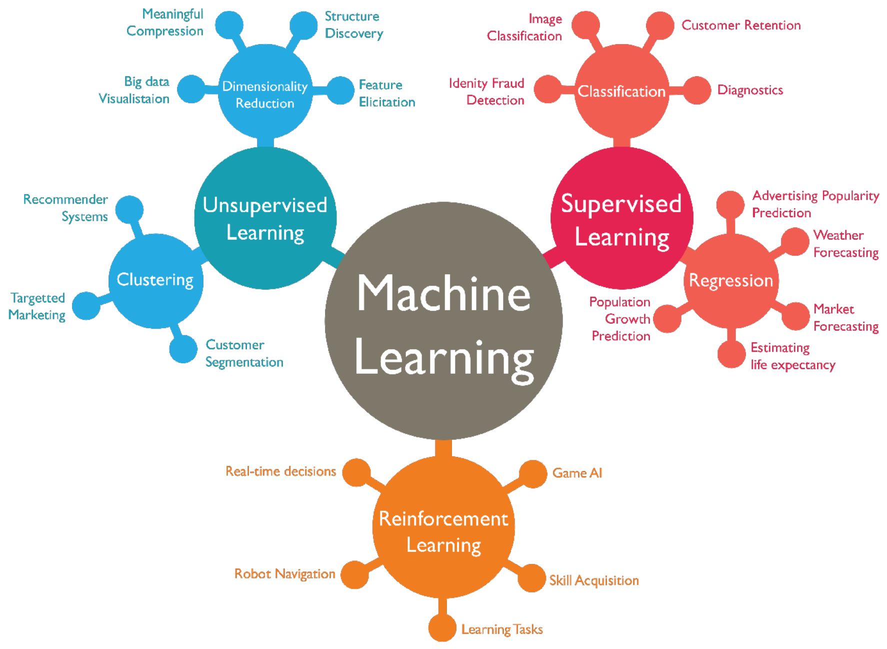


Figure 2.1: Classifications of Machine Learning (Lorberfeld 2019)

2.3.2 Procedure of Machine Learning Implementation

The procedure of implementing machine learning models require several steps of which can be divided into seven (Chollet 2018); collecting data, preparing data, selecting model, training, validating, tuning parameters and making predictions.

2.3.2.1 Step 1: Data Collection

Predictive models are only as good as the data from which they are built. Therefore, both the quantity and the quality of the data collected are crucial for developing high performance models.

2.3.2.2 Step 2: Preparing Data

Sometimes the data collected contains errors, noise, missing values, outliers, duplicates etc. To deal with this, the data needs to be cleaned and prepared in order to best fit the task at hand and the model that is going to be used. Additionally, since the order of the data should not affect what is being learned, the data is randomized.

2.3.2.3 Step 3: Model Selection

During the years, researchers have developed numerous of algorithms depending on what dataset and task is to be handled. Choosing the right model to deal with the desired task is crucial to obtain a good prediction.

2.3.2.4 Step 3: Training the Model

The goal of implementing a machine learning model, in this case, is to make a correct prediction as many times as possible. Depending on domain, data availability and amount, and data set particulars, the data set is split into two sets which is often called training and validation set. As a rule of thumb the training set is normally 80% and the validation set is 20%.

To exemplify, the formula of a straight line is

$$y = m \times x + b \tag{2.1}$$

where x is the input, m is the slope at that line, b is the y-intercept and y is the value of the line at position x. The values that are available for being adjusted, or as in the machine learning vocabulary *trained*, are m and b. Since there are many features, there might be many different m's in machine learning. The collection of the m values are formed into a weight matrix W, and the values b into a matrix of biases. The training process starts with some initial random variables for W and b, which is called hyperparameters[1], that attempts to predict the output with those values. By comparing the model's predictions to the actual output, the values of W and b is adjusted and corrected in order to aim for a better prediction for the output by the next iteration. This process is then repeated until the error is minimized.

As the theoretical error cannot be computed since the probability density function is unknown, a common method for minimizing an estimation of this theoretical error can be provided by cross validation. The criterion that needs to be minimized is the empirical risk, where Q is the cost function:

$$R_{emp}\left(\delta_f, S\right) = \frac{1}{m}\sum_{k=1}^{m} Q\left(\delta_f\left(\boldsymbol{\omega} x_k\right), y_k\right) \tag{2.2}$$

2.3.2.5 Step 5: Validating the Model

Once the training process is done, it is time to check if the model that was found is good. This is done by applying it to the percentage of the data set that was set aside as a validation set (here the suggested was 20%) which is used to assess the model performance. This validation set provides an unbiased evaluation of model fit on the training set while tuning the models hyperparameters. This unseen data is meant to be representative of how the model might performance when exposed to real world data.

2.3.2.6 Step 6: Hyperparameter Tuning

By tuning the parameters it is possible to see if the training of the model can be further improved. For more complex models the initial conditions (where in the exemplification above was W and b) can play a significant role in when determining the outcome of training. Knowing what initial values to use can be hard, and hence there are many approaches for optimizing these parameters which will not be discussed in this project.

2.3.2.7 Step 7: Making Predictions

The theoretical error obtained on the validation set can be biased since the validation set is used to select the final model. By this, it might be preferable to assess the overall performance of the model, a result of the cross validation process by using a third data set that has not been used either in the training set nor in the validation set.

Finally, when the parameters have been tuned and the model has obtained the best fit and overall performance for the data set, predictions can be made.

[1]Hyperparameter: A parameter whose value is set before the learning process begins

2.3.3 Approaches

There are many exciting developments in the area of ML. For instance in the manufacturing domain, new developments of computer science and the availability of software tools offers great potential (Wuest et al. 2016). However, it has been perceived that vast amount of information can propose challenges and may have negative impact as it can mislead in the terms of leading to wrong conclusions or distract from main issues. In addition, the field of ML is very broad with many different algorithms, methods and theories available. The types of machine learning algorithms differ in their approach, the type of data they input and output, and the type of problem they intend to solve (Susto et al. 2014).

Figure 2.1 shows an overview of the different classifications in machine learning, as well as some examples within the different classifications. In general, ML can be divided into three classes (Monostori 2003):

i. *Supervised learning* - where the algorithm build a mathematical model of a data set that contains both input and the desired output.

ii. *Unsupervised learning* – where the algorithm build a mathematical model based only on inputs from a data set.

iii. *Reinforcement learning* – where the algorithm build a mathematical model based on training information provided by the environment.

2.3.3.1 Supervised Learning

Supervised Learning (SL) is used for the majority of practical machine learning problems (Brownlee 2016). Due to the availability of expert feedback and labeled instances, SL is learning from examples provided by an external supervisor, hence supervised. “Supervised Learning is a machine learning paradigm for acquiring the input-output relationship information of a system based on a given set of paired input-output training samples” (Q. Liu and Wu 2012, p. 1).

The goal is to approximate the mapping function based on input data (X) well enough to be able to predict the output variables (Y).

$$Y = f(X)$$

The learning process stops when the algorithm reaches a desired level of performance.

SL is generally divided into two groups; *regression-* and *classification problems*. In a regression problem the output variable is a continuous, real value e.g. a number. In classification problems the output variable is a discrete category e.g. a color.
Some examples of supervised learning (Ensari et al. 2019):

- Linear Regression
- Logistic Regression
- Decision Trees (DT)
- K-Nearest Neighbors (KNN)
- Naïve Bayes (NB)
- Random Forest (RF)
- Support Vector Machines (SVM)

2.3.3.2 Unsupervised Learning

Unsupervised (USL) is where you only have input data (X) and no output variable (Brownlee 2016). In other words, there is no feedback of identified output from an external expert, and the algorithm itself is supposed to identify patterns from existing data. *Clustering* is one group of USL, where the aim is to discover inherent groupings in the data, e.g. the purchasing behaviour of people. The other category of USL is *association*. Here, the aim is to discover rules that describe large portions of the data e.g. if people buy bread (A) they also tend to buy butter (B). Some examples of unsupervised learning (Ensari et al. 2019):

- K-means Clustering
- Hierarchical Clustering
- Singular Value Decomposition
- Nonnegative Matrix Factorization
- Hidden Markov Models

2.3.3.3 Reinforcement Learning

Reinforcement Learning (RL) is defined by the fact that there is less feedback, since it is not the proper action but only the evaluation of the chosen action that is given by the external expert (Monostori 2003). Some researchers define RL as a special form of SL (Pham and Afify 2005). Basically, RL is learning through interaction with the environment. By trying, instead of being told, the learner has to discover which actions generate the best result (Wuest et al. 2016). Experienced situation with high rewards tends to be repeated, while avoiding situations with low rewards (Q. Liu and Wu 2012).

2.3.4 Artificial Neural Networks

An artificial neural network is a computational model based on the structure of the neural network in the human brain. Generally, a neural network will consists of three different layers, the input layer, the output layer and the hidden layer. The input and output layer is nothing more than what the word indicates, the input and the output of the computational model. The interesting aspect of a neural network is what happens between these two layers, in the hidden layer, often refereed to as the black box layer. Number of hidden layers depends on the structure and complexity of the task to solve, and this also counts for the number of nodes. As a main rule one usually say that the number of nodes in the hidden layers should midway between the number of input and output nodes. One also wants the number of nodes in the hidden layer to be no larger than two times the number of input nodes. This is to avoid that the model over-fits to only match the training data, resulting in a computational model that will be unable to recognize any data patterns outside the training set. The neurons in the hidden layers take in a set of weighted inputs and produce an output with the help of an activation function. The activation function transforms the weighted input from one node to the output for that node, and is in other words a non-linear mapping between an input parameter to an output parameter. Figure 2.2 shows a potential structure for one node in a neural network, where the input is multiplied by a certain weighting rule, mapped to the output of the neuron through an activation function, creating the output of the neuron. In figure 2.3 a network of an input and output layer and two hidden layer is shown. Each node in the network has more or less the same structure as shown in figure 2.2, creating the basis structure for a neural network.

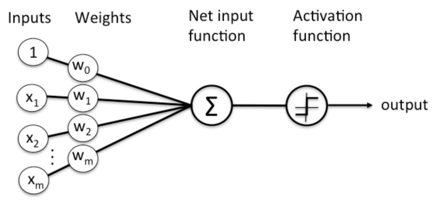


Figure 2.2: Possible Structure For One Neuron

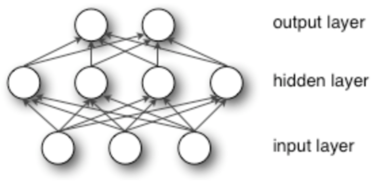


Figure 2.3: Neural Network Structure

Machine learning in condition monitoring for predicting maintenance decision making

2.3.4.1 Error in Artificial Neural Network

When evaluating the performance of a neural network, error of predictions compared to actual output values are calculated. This can be done by using various error functions or performance metrics. Choice of error function used during training is typically determined by the type of output unit. Typical measures used are for example Sum of Squares Error, Mean Squared Error or Mean Squared Prediction Error, when dealing with regression problems. For classification problems Cross Entropy Loss is the most common measure for error evaluation.

Neural Networks are trained by dividing the dataset into a training set and a testing set. The model is trained using the training data, and later the performance of the trained model is evaluated. For any machine learning algorithm, the "bias-variance dilemma" is highly relevant, and involves finding a balance between the accuracy and the generalization of a model. An example of such a trade off is that a model can perform exceptionally well on training data, while showing poor accuracy when applied to a test set. This concept is called over-fitting, and happens when a model too well represents the training data, hence it will fail to generalize well on data outside the training set, characterized by a large variance. Hence over-fitting model to the testing set will result in a test error converging to zero, while the test error will increase drastically. Such an error indicates that the model is not capable of making accurate predictions for observations not previously seen in training. The bias represents the inability of the physical model to accurately approximate the function, known as the approximation error and the variance represents the inadequacy of the empirical knowledge in the training sample, known as estimation error. There is a trade off between these two, typically known as the loss function.

$$\text{Loss function} = (Bias)^2 + Variance \tag{2.3}$$

The model with the optimal predictive capability is the one that leads to the best balance between bias and variance. Both can only be eliminated in the case when the number of training samples becomes infinitely large, which is achievable in prat ice.

2.3.5 Recurrent Neural Networks

Recurrent neural networks (RNN) is a type of neural network that make use of the sequential information that the input may have (P. Liang and Bose 1996). This is in contrast to the traditional acyclic feed forward neural network, which assumes that all inputs are independent. This assumption is fine for many tasks but for problems like sentence sentiment analysis or hand written text recognition there is a lot of information in the sequence of the input. A recurrent neural network uses an internal state to utilize the sequential information present. Figure 2.4 shows how a generic RNN is constructed. As the picture shows, the network actually performs the same operation for every element in the series. Information of the previous element is transferred, "remembered" and used in the calculations for the next element and so on in the sequence.

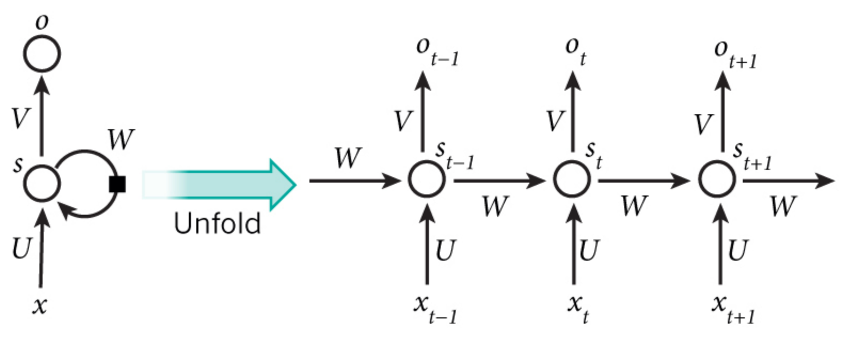


Figure 2.4: RNN unfolding architecture

Figure 2.4 shows how the very simple RNN would look if unfolded. Here x is the input and s is the memory of the network and is called the hidden state. The hidden state is calculated based on the previous hidden state and input combined through the activation function. o is referring to the output and in many cases only the last output is of interest like in sentence sentiment analysis. It is worth noting that these basic RNN have short memories. As a answer to this the long short-term memory RNN was developed (Hochreiter and Schmidhuber 1997).

In theory, RNN can use the information from an arbitrarily long input sequence, however in practise RNNs are only able to remember a few steps back in the sequence (P. Liang and Bose 1996). Figure 2.5 shows how the information from the first input ("What") in dark green is all most forgotten as input "05" is about to be processed. This is because the activation functions can take large values and squeezes it to be in between 0 and one or minus one and one. This effect means that the previous steps get multiplied by values between zero and one and therefore gets smaller and smaller for each cell it has to pass.

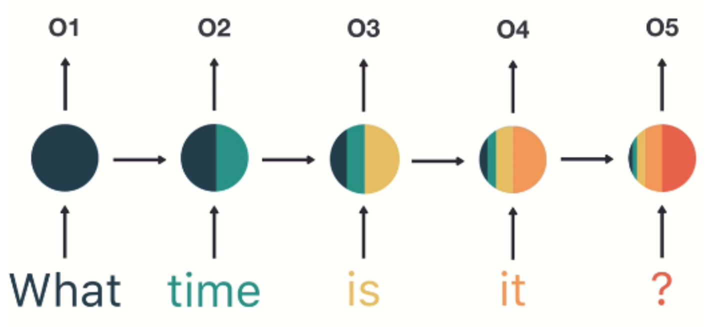


Figure 2.5: Shows how information is forgotten in a recurrent neural network

RNN normally forms big and complex structures because they are recurrent. This helps in complex sequence recognition, but may cause problems with learning. This is because of the fact that as backpropagation is preformed the gradient gets either exponentially smaller or bigger as the error is fed back in the network. And because RNNs typically have large and complex structures the error is particularly exposed to vanishing gradient. It means that the first layers in the RNN does almost no learning (Hochreiter 1998).

2.3.5.1 Long Short-Term Memory

Long short-term memory (LSTM) neural networks are a particular type of RNNs. They where developed as a response to the problem of short memory caused by the cell structure in traditional RNNs (Hochreiter and Schmidhuber 1997). They work in a similar way to the basic RNN discussed above. the difference is the information flow within the LSTM cells as depicted in figure 2.6.

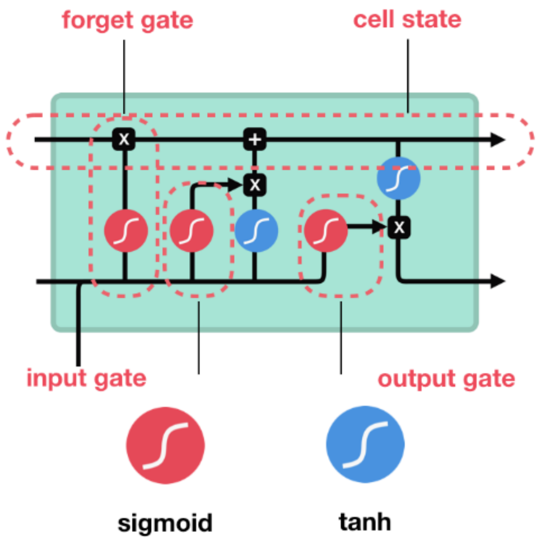


Figure 2.6: LSTM internal scheme

The core concept of the LSTM network is in the cell state and how it uses different gates to regulate the flow of information. The cell state will for a LSTM network act as it's memory and is implemented in such a way that information from earlier steps in the time series can make it all the way to later steps. This is how LSTM network alleviates the "short-term" memory problems of a simple RNN. As the cell state is passed to the next cell, this cell will evaluate how much information to forget or retain from the previous cell state. The forget gate is responsible for evaluating how much information to let through and be part of the new cell state. The information let through the forget gate gets supplemented by the new cell state in the current node. The forget gate in it self can be a neural network and is therefore able to learn what is relevant information and what is not (F. A. Gers, Schmidhuber, and Cummins 1999).

2.3.5.2 GRU

The Gated Recurrent Unit or GRU for short, is comparable to the LSTM network as they both uses gates to prevent vanishing gradient and control information flow from previous steps. The difference is that the GRU only has two gates, the reset gate and an update gate (Fu, Z. Zhang, and L. Li 2016), while the LSTM has three gates, namely input, output and forget gate. The GRU unit is relatively new but is fast becoming popular due to the fact that it is computationally cheaper and because its architecture is simpler and the network are in some cases faster to train. Although it is worth mentioning that LSTM networks in theory have a longer memory and therefore should outperform the GRU based network in memory intensive classification tasks with lots of data.

Chapter 3

Literature Review

3.1 Systematic Literature Review

3.1.1 Literature Review Planning Protocol

Research Questions

The literature review was conducted with the aim to answer the following research questions:

- **Q1**: What machine learning approaches are most commonly used in PdM?
- **Q2**: To which and what extent has ML approaches been used in anomaly detection, diagnostics and prognostics, respectively?
- **Q3**: Which fields are the less researched, and what are the challenges?
- **Q4**: To what extent can ML-approaches be used as a solid ground for decision making regarding maintenance strategies and provide a good uncertainty quantification?

Database for literature searching

The study was conducted with the use of two literature databases, Google Scholar and ORIA. The latter is a common database with all of the available material at most Norwegian academic and research libraries, whereas the former is a freely web search engine that indexes scholar literature.

Exclusion criteria

In the litterateur review, work not related to predictive maintenance and machine learning approaches have been excluded, as well as work dated before 2010, with the emphasis on the most recent published work.

3.1.2 Execution

The choice of keywords for the search strings was based on terms commonly found in the literature and terms related to this review. The first research question contained the following strings of keywords:

- ORIA: [Title contains predictive maintenance and title contains machine learning]
  Which gave 90 results.

- Google Scholar: [allintitle: predictive maintenance machine learning]
  Which gave 66 results.

Looking into the different research fields of machine learning and predictive approaches the following keyword strings were executed:
Prognosis

- ORIA: [Title contains Predictive Maintenance and all fields contain Machine Learning and all fields contain Prognosis or Title contains Predictive Maintenance and all fields contain Machine Learning and all fields contain Prognostics]
  Which gave 66 results

Diagnosis

- ORIA: [Title contains Predictive Maintenance and all fields contain Machine Learning and all fields contain Diagnosis or Title contains Predictive Maintenance and all fields contain Machine Learning and all fields contain Diagnostics]
  Which gave 100 results

Anomaly Detection

- ORIA: [Title contains Predictive Maintenance and all fields contain Machine Learning and all fields contain Anomaly Detection or Title contains Predictive Maintenance and all fields contain Machine Learning and all fields contain Failure Detection]
  Which gave 103 results

The execution was conducted in the time domain October-November 2019.

3.1.3 Selection

After the search was executed the papers were selected based on a correlation between their publication date, their relevance after reading the abstraction and the amount of times the paper had been cited by other researchers.

3.1.4 Publications and trends

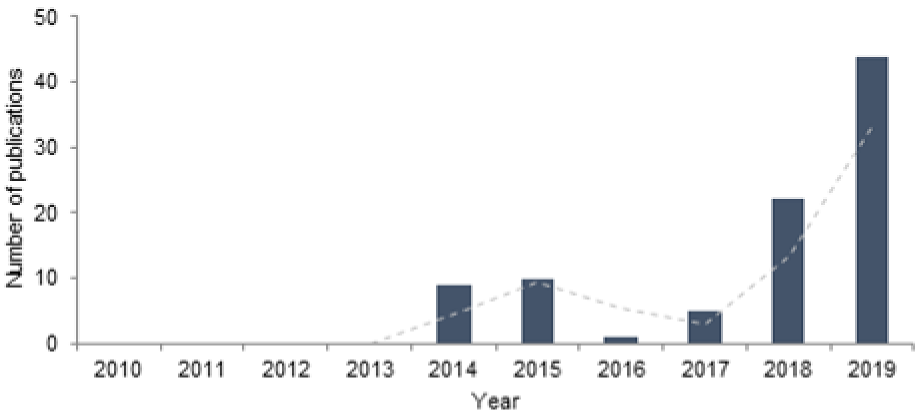


Figure 3.1: Number of papers published per year

As seen in figure 3.1 there has been an emerging trend in the research field regarding the use of machine learning to create data-driven models used for predictive maintenance. As expected, diagnosis and anomaly detection are the two fields where it has been done most research, whereas less has been done on prognosis. This can been seen in figure 3.2

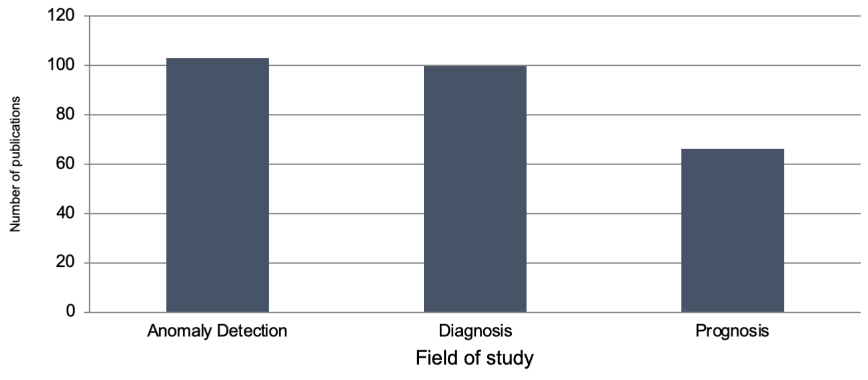


Figure 3.2: Number of papers published in each field of study

3.2 Machine Learning Used for Predictive Maintenance

3.2.1 Anomaly Detection

The goal of anomaly detection applied in maintenance is to create a system that provides early warnings of potential system failure (Rabatel, Bringay, and Poncelet 2011). With this follows several challenges like minimizing false positives and negatives, as well as having a model which is adaptable to environmental changes. The latter involves any system configuration that changes the definition of normal behaviour.

3.2.1.1 Applications in Maintenance

Time series anomaly detection has been applied to both streaming and batched-sized data, and the following chapter reviews some of the most developed anomaly detection models using machine learning.

Basora, Olive, and Dubot (2019) reviews some of the recent anomaly detection systems applied in air traffic operation and preventive maintenance. The referred system belongs to NASA which stores monitored data from major airlines, where the the goal is to automatically detect anomalous events based on on-board historical data. Basora, Olive, and Dubot (2019) explains that the anomaly detection system is limited to exceedance detection algorithms. This means that an event is only considered an anomaly if it exceeds a predefined threshold. This approach is the most developed form of anomaly detection systems, but it has it's shortcomings. Accurately defining the threshold is crucial in order to avoid both false positives and false negatives. This threshold is not generalizable to multiple systems and needs to be determined by domain experts. In addition, this approach is not robust enough to handle all possible events that could occur, due to the biased approach of configuring predefined limits.

Rabatel, Bringay, and Poncelet (2011) reviews the use of anomaly detection in preventive train maintenance. The purpose of anomaly detection is to detect abnormal behaviour, which naturally demands a clear definition of what normal behaviour is. This is a challenging task, since the normal behaviour is greatly context dependent. Real life systems are often having multiple operational modes, resulting in multiple types of normal behaviour. A well developed anomaly detection system must therefore be able to adapt to the dynamic changes of the environment. Not having this property will result in many false positives. To illustrate this, an example is proposed by Rabatel, Bringay, and Poncelet (2011). Given some condition monitoring on trains, the sensor values will vary dependent on the shape of the railroad. During a journey the train will experience turns and both uphill and downhill slopes, which can be considered as different operational modes. First the train moves straight forward, and a normal behaviour is determined based on historical data. Then it is exposed to a turn, and a spike in the sensors occurs. If the anomaly detection model has not determined the normal behaviour for each possible operational mode, it will provide false positives.

In order to overcome this challenge, Rabatel, Bringay, and Poncelet (2011) proposes a two step approach. It begins with using some unsupervised data mining technique, like SVM (Shi 2011) or clustering algorithms, in order to extract essential information from the dataset. This information includes identifying the different operational modes and describing their normal behaviour by determining upper and lower bounds for the sensor values. When performing this sensor datamining step, its critical that the data is not contaminated with outliers in order to correctly classify normal behaviour. One way to make sure that the data does not contain any false information, is to fit it into a suitable mathematical function prior to feeding it into any machine learning algorithm (Elasha et al. 2019). The next step is detecting anomalies based on the trained model. Given a new data instance and the systems' current operational mode, determine it's behaviour as either abnormal or normal according to the calculated boundaries of normal behaviour. This two-step approach can be performed by a range of different machine learning techniques like, SVM (K.-L. Li et al. 2003) and clustering (Peng and Z. Liang 2018).

Peng and Z. Liang (2018) proposes a strategy for anomaly detection with changing cluster centers (ADCCC). Initially, the dataset is processed with a k-means unsupervised algorithm (Krishna and Murty 1999). This data mining technique locates k groupings of the dataset. It begins with randomly determining k centroids, before it iterates over the dataset until the positions of each k clusters are optimized. An outlier degree is then calculated for new samples that enters the system. Given a new sample x, calculate the Euclidean distance between x and all the cluster center positions. Add x to the closest cluster C, and recalculate the new center for that particular cluster. Further, calculate the relative distance, d_i, between the original cluster,C (without the new sample), and the new cluster, C' (with the new sample).

$$d_i = dis(C, C')/d_q, \tag{3.1}$$

where d_q represents the maximum distortion of the cluster center. If the value of d_i is greater than 1, x its considered an outlier sample. The next step is to determine some threshold τ that separates normal and abnormal behaviour. Similar to Rabatel, Bringay, and Poncelet (2011) this determination requires samples without outliers. Determining the threshold is a challenging task, where a too low threshold results in many false positives alarms, and a too high threshold results in many false negative alarms. Peng and Z. Liang (2018) determines the boundaries by some percentage of a normal sample's outlier degree. New datapoints can now be judged as either normal or abnormal by calculating its outlier degree d_i. If d_i is larger then the abnormal threshold, then the new sample is considered abnormal and an early warning is provided.

3.2.2 Diagnostics & Prognostics

In recent years machine learning techniques have been vastly used in diagnostics and prognostics of system components. Continuously improving diagnostics creates a framework for the end goal of predictive maintenance, damage prognosis and decision making. Farrar and Lieven (2006) defines damage prognosis as the estimated remaining useful life of a system. This estimate is based on three parameters; a quantified threshold describing when the system have failed and a corrective maintenance is necessary, a diagnostic part assessing the system current damage level, and forecasting models predicting possible future loads that the system will experience. These parameters would also have to contain original assumptions regarding system's loading compared to the actual experienced load, as well as any information regarding completed maintenance or system testing (Farrar and Worden 2012). Any information concerning "user feel" about how the system is operating should also be incorporated.

Damage prognosis (DP) is still an evolving technology with few deployed applications. This chapter illustrates the current state of damage prognosis by reviewing the most established DP systems and discussing its characteristics. Then, reviewing some emerging technologies that might improve the damage prognosis, and concluding with future development.

3.2.2.1 Time Series Analysis & Forecasting

Being able to predict the future is a big competitive advantage for any industry and business. Time series analysis and forecasting has been an active research area over the last decades, and there is still being put a lot of effort into developing methods that can contribute to decision-making by performing time series analysis and forecasting. The accuracy of these methods is crucial and hence the research of improving the precision and effectiveness of forecasting models has never stopped (G. P. Zhang 2003). In this chapter, some of the methods that has been developed and tested is reviewed, and the different algorithms is discussed and compared.

3.2.2.1.1 Time Series Models and Applications

In 1970, George Box and Gwilym Jenkins published their book *Time series analysis, forecasting and control*, where they proposed the Box-Jenkings method. For nonstationary time series, the approach starts with the assumption that the process that generated the time series can be approximated by the classical model Autoregressive Integrated Moving Average (ARIMA) (Box et al. 2015). Classical models has the advantage of being easy accessible, and during the past four decades ARIMA has been the most popular method in time series forecasting. The model describes how the dynamics of the time series behave over time, and simply continues the model dynamics into the future.

Since the ARIMA model has been widely used in the forecasting research and practice, comparing ARIMA to different deep learning models has a broad area of application. Particularly many studies have been applied to financial time series forecasting and the results show that ANN models outperform ARIMA models. This is also supported by a case study on Iranian poultry retail price done by Karbasi, Laskukalayeh, and Seiad Mohammad Fahimifard (2009) and a study on stock data obtained from New York Stock Exchange, where the findings support this claim (Adebiyi, Adewumi, and Ayo 2014).

As discussed above, ARIMA and ANN are often compared with mixed conclusions depending on the superiority of prediction and forecasting performance. Several researchers such as H. Liu, Tian, and Y.-f. Li (2012), Cadenas and Rivera (2010) and Chen and Lai (2011) suggests a hybrid of ANN and ARIMA in order to forecast wind speed. Because it can be hard to know the characteristics of the data in a real life problem, taking advantages of the strengths of ARIMA and ANN models in linear and nonlinear modeling respectively can help solving this issue. G. P. Zhang (2003) uses three well-known data sets (the Wolf's sunspot data, the Canadian lynx and the British pound/US dollar exchange rate data) to demonstrate the effectiveness of the hybrid method. By comparing the results given by the hybrid to the results given by using ARIMA and ANN separately, the conclusion stated that hybrid models predict with a higher accuracy.

Even though neural network often outperforms classical models, they are not always superior. S. Fahimifard et al. (2009) reviews the need of exchange rate forecasting in order to prevent its disruptive movements by comparing several models (GARCH, ARIMA, ANN, ANFIS). Briefly, the conclusion was that ANFIS model had an effective result in forecasting the accuracy of exchange rate. In financial time series, many studies have been using ANNs (G. Zhang, Patuwo, and M. Y. Hu 1998). Nevertheless, Kim (2003), F. E. Tay and L. Cao (2001) and L.-J. Cao and F. E. H. Tay (2003) concludes that SVMs provide a promising alternative for financial time series forecasting.

A study done by Voyant et al. (2017) compares a selection of papers concerning global solar radiation prediction combining ML methods. In this paper it can be seen that compared to classical regression models, papers using SVR, ANN, k-NN, regression tree, boosting, bagging or random forest give better results. In terms of prediction, SVM seems to be easier to use than ANN (since the optimization step is quite complex in ANN) but they tend to give the same results. The use of regression trees and similar methods give excellent results, but is rarely used.

3.2.2.2 Diagnostics & Prognostics on Sensory Data

3.2.2.2.1 Rotating Machinery Applications

So far, the most developed and successful condition monitoring and damage detection systems are applied to rotating machinery (Farrar and Worden 2012). Rotating components are heavily used in many industries (Senapaty and Rao 2018), and it is therefore a greatly researched area.

Condition monitoring of rotating machinery is mainly based on measured vibration (Senapaty and Rao 2018). Vibration based condition monitoring began with a simple procedure by holding a screwdriver on the machine and listening after some change of the acoustic signal, as well as feeling the change of vibration response from the screwdriver (Farrar and Lieven 2006). Today's system applies more advanced data acquisition techniques, measuring the key characteristics of vibration; amplitude, frequency and phase, and extracting the most relevant features (Senapaty and Rao 2018). The system health can then be evaluated by recognizing patterns of change in the vibration signal.

There are many machine learning techniques applied in diagnostics and damage prognosis of rotating machinery. Mahamad, Saon, and Hiyama (2010) proposes the use of Artificial Neural Networks (ANN) in order to improve the RUL prediction. The dataset applied to the algorithm consists of measured vibration from the machine. Further, key features such as the mean, round-mean-squared (RMS) and kurtosis are extracted from the data and fed into a feed-forward neural network (FFNN). The FFNN is a three-layered neural network with an input-layer, one hidden layer and an output layer, as shown in 3.3.

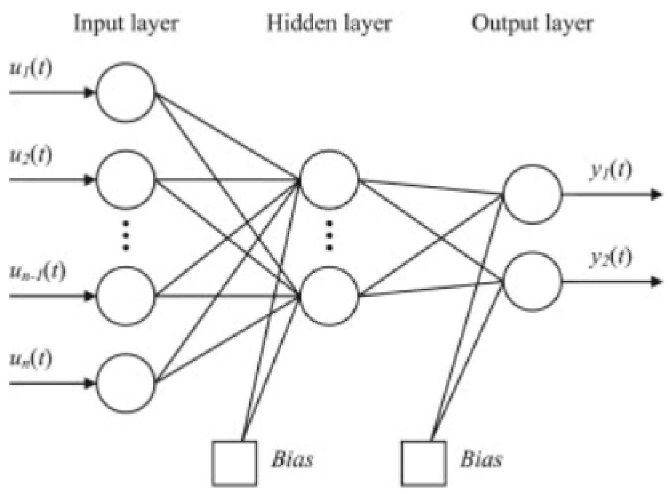


Figure 3.3: Feed-forward neural network. (Mahamad, Saon, and Hiyama 2010)

The goal is to calculate the RUL of a bearing on a rotating shaft driven by an AC motor. Each bearing has two attached accelerometers, one horizontal and one vertical. The vibration sampling rate is 20kHz and is collected for one second every 10 minutes, resulting in one inspection every 10 minutes with 20480 data points. The bearings are measured until failure, which is after seven days. Prior to feeding the data to the FFNN, the datapoints are fitted with the Weibull hazard rate function. This is done to remove external noise in the dataset.

Six inputs are fed into the FFNN. Two of them, t_i and t_{i-1} represents the time stamps of the current and previous inspection. z_i^1 and z_{i-1}^1 represents the RMS value of the current and previous inspection, while z_i^2 and z_{i-1}^2 represents the kurtosis values. The output of the FFNN is the normalized life percentage at inspection i, which is proportional to time. A 100% output value represents that the bearing has completely failed. A cross-validation technique is used while training the model, and mean-squared-error (MSE) is used to evaluate the performance of the algorithm. The results shows an error of $9.99e^{-13}$ for the training set and $3.1e^{-12}$ for the validation set.

Elasha et al. (2019) performs a similar RUL estimation by using a combination of a regression model and a multi layered artificial neural network. The experiment is also performed on a bearing structure, this time within a gearbox of a wind turbine. The acquired data is measured vibration from accelerometers on the bearing, and the key features extracted are RMS, kurtosis and an energy index. Data was collected every 10 minutes for 50 consecutive days, with a sampling rate of 97656 Hz for 6 seconds. Similar to Senapaty and Rao (2018), the raw measured data could not be used as input in the prediction models due external noise and variation of operational conditions. Therefore, the data was first fitted with an appropriate mathematical model.

The regression model was first applied to the dataset in order to obtain the relation between condition indicators and time (Elasha et al. 2019). The reason for using the regression model prior to feeding the dataset into the ANN, was to determine which feature (RMS, kurtosis, energy index) that was the best indicator of degradation. A feed-forward back-propagation artificial neural network was trained using the Levenberg-Marquardt algorithm which has proven to outperform gradient descent methods(Mukherjee and Routroy 2012). The ANN was structured with an input layer consisting of two inputs, the RMS and the kurtosis, two hidden layers and one output layer. To estimate the RUL, the output of the ANN is extrapolated to a predefined threshold. The actual RUL is given by the difference in time at the current inspection and the final inspection. The last recorded inspection is assumed to represent a failure. The prediction error is obtained by using sum of squared error(SSE) between the actual RUL and the predicted RUL.

3.2.2.2.2 Civil Engineering - Bridge Structures

Structural health monitoring (SHM) is highly applied on bridges around the world, like Tsing Ma bridge, Golden gate bridge and Confederation bridge (Chang 2013). Ko and Ni (2005) explored different data acquisition techniques and types of sensors used in SHM on multiple bridges around the world. The types of sensors includes; corrosion sensor, temperature, strain, displacement transducer and hygrometer.

Chang (2013) performed an experiment on bridge health assessment and damage localization on one of Sydney's largest bridges. Sensors were placed on six different joints, were one of the joints were considered damaged. The idea was to use a support vector machine in order to distinguish faulty joints from normal joints. Similar to Mahamad, Saon, and Hiyama (2010) and Elasha et al. (2019), the data collected was vibration based. Vibration is caused by vehicles passing the location where the sensors are placed. It is recorded by using a tri-axial accelerometer placed in each on the joints and is sampled with a frequency of 400Hz. In order to use a supervised machine learning technique like the SVM, labelled data is required. The data is labeled in a way that 5 of the measured joints are considered having normal condition, while one of the joints are damaged.

Chang (2013) uses a binary Support Vector Machine for the supervised training, and a one-class unsupervised Support Vector Machine for anomaly detection. A 600 points time series is extracted by each sensor on each recorded joint, giving a total of 1800 data points per joint every inspection. While training, the label of each inspection is known. The SVM is therefore fed with datapoints from each sensor with corresponding label. The feature weights are continuously adapted when new inspection data enters the algorithm, by minimizing the calculated error. To reduce outliers a one-class unsupervised SVM is used for anomaly detection. This algorithm assumes that there exist a strong pattern between the feature within the positive examples, while the negative examples can have many different properties (Erfani et al. 2016). This is done by applying a kernal function that maps the input space to a higher dimensional feature space, with the goal of creating a more convincing separation of the normal data and outliers (Erfani et al. 2016).

By combining the two versions of SVM and a 10-fold cross validation, Chang (2013) performed five iterations on three individual datasets. Dataset 1 contained inspection data from two normal joints and one damaged joint. Dataset 2 had data concerning all joints, and dataset 3 contained data from only one sensor from all 6 joints. Inspecting the result from each individual algorithm shows that dataset 2 had the highest accuracy with 99.71% and 99.30% accuracy for the time domain and the frequency domain respectively. Dataset 3, containing only one sensor from every joints outperformed dataset 1 with all sensor from three joints, proving that having multiple examples of damaged vs normal, but less data (dataset 3) is more accurate than having more data per joint, but less examples of different classifications.

3.2.2.3 Combining Multiple Machine Learning Models - Ensemble Method

Intuitively, the approach when performing data-driven prognostics is to create different machine learning models. Each of these models are individually trained on the dataset and evaluated. The most accurate model is then chosen. C. Hu et al. (2012) explains two shortcomings with this approach; (i) the chosen model with the highest accuracy might not be robust enough on its own; (ii) every discarded algorithms leads to waste of resources. C. Hu et al. (2012) proposes an ensemble data-driven approach in order to overcome these challenges.

C. Hu et al. (2012) is combining five different data-driven prognostic algorithms in his ensemble approach:

- Model 1: A similarity-based interpolation (SBI) approach with a relevance vector machine (RVM) as the regression technique.
- Model 2: SBI with a support vector machine (SVM) as the regression technique
- Model 3: SBI with least-square exponential fitting
- Model 4: A Bayesian linear regression approach with least-square quadratic fitting
- Model 5: A Reccurent Neural Network (RNN)

In order to perform a SBI RUL prediction two procedures are required, namely (i) identifying initial health condition; (ii) the actual RUL calculation. The determination of initial health condition is based on the predicted degradation curves allocated using the regression techniques mentioned above. Given a training set, each of the regression methods can compute multiple degradation curves. During the initial health condition calculations of the RUL prediction, each of the computed curves are compared with the test data. The curve with the least sum of squared difference (SSD) is selected, and corresponding initial health condition is determined (Wang et al. 2008). Finally the RUL prediction is calculated by the difference of the total life of the most similar curve and the current life of the targeted unit. Model 1-3 follows the above-mentioned approach but with different curve-fitting techniques.

The fourth model is following an extrapolation based approach, which does not make a comparison between training and test data as the SBI does, but rather obtaining the prior distribution of degradation with respect to the parameters. Further the testing data is applied to update the degradation model before extrapolating a RUL prediction based on a given threshold (Coble and Hines 2008).

The fifth model is a artificial neural network with four layers (two hidden, input and output) with similar configuration as Mahamad, Saon, and Hiyama (2010) presented above.

The next step of an ensemble method is combining the different algorithms into one stronger predictor (C. Zhang and Ma 2012). The idea is to use a weighted sum of all the models, where the modelweights is assigned according to the performance of the model. Models with high performance get greater weights which makes sure that the best performing models have the highest impact on the final prediction. The assigning of weights can be done with three different techniques (C. Hu et al. 2012).

- Accuracy-based weighting, is only concerned with the accuracy of each of the models. The accuracy, ϵ, is calculated by the model error. For j different models, each weight, w_j is then calculated by:

$$w_j = \frac{(\epsilon^j)^{-1}}{\sum_{i=1}^{M}(\epsilon^j)^{-1}}, \tag{3.2}$$

  where M equals the number of algorithms in the ensemble (C. Hu et al. 2012).

- Diversity-based weighting assigns weight to the algorithms dependent on their prediction diversity.

- Optimization-based weighting, is an optimization problem which combines both accuracy and diversity, resulting in high prediction accuracy and robustness (C. Hu et al. 2012).

The final ensemble contains the flexibility and strengths of each individual prognostic algorithm, to create a more robust system with higher accuracy, reduced bias and more generalize to handle different condition monitoring problems.

3.2.2.4 Diagnosis & Prognosis on Event Log Data

To be able to create models that can be used for predictive maintenance, insight into the running condition of the equipment is required. Knowledge about the equipment can be achieved by adding sensors and detectors for monitoring, which is an effective solution. Then again, for in-service equipment this might be infeasible, due to the installation of sensors on the running hardware. A lot of research has been done on the ability to gain insight into the workings of equipment by only studying it logs.

In theory, one can trace back how a piece of equipment was used by analyzing it logs and using this information to detect potential issues. Log-data in prognostics and diagnostics begins with filter out a large amount of noise, extract predictive features and collect the known cases for learning and evaluating models (Sipos et al. 2014).

Log-data is a collection of events recorded by different applications running on the equipment. Such a log can be viewed both as a symbolic sequence and as a numeric sequence, with the possibility to include other data sources such as unstructured data and categorical features. Using log-data for predictive maintenance is an old concept, where patterns were detected manually, which is a time consuming process, and requires a lot of experience.

3.2.2.5 PdM using Log-Data

Predictive maintenance (PdM) strategies apply prognostic models to be able to forecast the equipment condition. Such a model is a combination of multiple predictors which aims to calculate the risk of a specific endpoint, and predict the equipment's health, probability of failure and remaining useful lifetime (RUL) (Steyerberg et al. 2013). Log-based approaches differs from sensor-based approaches because the models are only trained using historical event log-data, instead of sensor data. Event logs consist of time-ordered events. According to the research paper by Korvesis (2017) a event is defined in the following manner:

- An event is composed by a timestamp and other attributes that contain information about the event
- The attributes might include a system or subsystem identifier that generated the event
- The attributes might be a task id, description about the activity, failure description and an unique event identifier.

(Korvesis 2017).
The function of a log-based model is to give a warning a certain period before a breakdown and will work as an early monitoring system rather than a continuous monitoring system (Gutschi et al. 2019).

A research conducted by Fink, Zio, and Weidmann (2015) predicted the remaining useful life (RUL), based on Discrete-Event Diagnostic Data. They defined the RUL as the following: "RUL is defined by the time interval between the instance when the impending failure condition is detected, isolated and identified and the time when the failure occurs" (Vachtsevanos et al. 2006, p. 160). The RUL is defined as the time interval between the current age of the component, t, and the predicted time to failure, T, given the past condition profile up to the current time $Z(t)$. In their research the problem of predicting the rule is classified as a binary classification task, which is the simplest way to classify a machine learning problem. The goal in such a task is to classify the data points into one of two bags - positive or negative. Thus, in their research two classes needed to be defined:

- A class of system conditions which leads to a failure after a defined time interval
- A class of system conditions which remain in healthy condition after the defined time interval

They used a fixed length RUL in their approach, t_{pred} which is defined by the user. The predefined RUL needs to be at least as long as the minimum time required by the maintenance team to plan and prepare the maintenance actions. In addition, they define a second time interval within the failure is predicted to occur t_{fail}. The binary classification problem considered 255 distinct event codes for a brake system for a train, over an analysis period of 313 days. The undemanded service brake in railway vehicles was the event selected to predict the RUL. A pattern of recorded parameters belonged to the positive class if within the time period t_{fail} at least one disruption even occurred, and if no operational disruption even occurred the time pattern was classified as negative. In their research they use Extreme Learning Machines (ELMs) which are feed forward networks with a single hidden layer. ELM is known to combine the strengths of several machine learning techniques, such as SVM and feed forward neural networks. The major advantages of ELMs are that they are computationally efficient and do not require manual parameter tuning, which is the case for a lot of the other machine learning approaches. SVM is known to be a popular approach, in a research by Huang, Ding, and H. Zhou (2010) the two methods were compared for a standard classification problem, and the author concluded that "ELM and SVM for classification are equivalent, but ELM has less optimization constraints due to its special separability feature" (Huang, Ding, and H. Zhou 2010).

However, the main drawback of the ELMs is its sensitivity to ill-conditioned data, meaning that a small change in the input will lead to a large change in the output (Yildirim and Özkale 2019). Thus, in their research they combine ELMs with Ridged Regression which is an approach known to overcome this sensitivity. This approach gave good results, where 98.20% of all of the patterns from the positive class was correctly discriminated from the negative ones. Fink, Zio, and Weidmann (2015) also processed the input patterns using a Multi-layer Perceptron (MLP), which gave a miscalculation rate of 50%. Their assumption was that such MLP algorithms require additional pre-processing and use of feature extraction algorithms, which was not used in the pre-processing phase. One can argue that this method has a major drawback due to the fact that only a RUL with a predefined length can be predicted.
A similar approach was conducted by Sipos et al. (2014) on predicting failures in medical scanners. In their approach, a target component and a collection of historical log and service data was used to formulate the problem. Similar to Fink, Zio, and Weidmann (2015) their approach was to construct a binary classifier to predicting failures, and their problem was viewed as a Multiple Instance Learning (MIL) problem.

In a standard supervised approach the training data consists of instances such as $(x_1, x_2, ..., x_n)$, and the labels $(y_1, y_2, ..., y_n)$ where $x_i \in \chi$ and $y_i \in Y$. The goal is essentially to train a classifier function, $h(\chi) : \chi \rightarrow Y$. In a MIL problem on the other hand, the training set consists of bags $(X_1, X_2, ..., X_n)$, where $X_i = (x_{i1}, x_{i2}, ..x_{im})$ and the bag labels $(y_1, y_2, ..y_n)$, where $x_{ij} \in \chi$ and $y_i \in 0, 1$ Hence a MIL approach aims to label a set or bag of input, by restricting labels to be binary, the assumption is that every bag is positive if it has one positive input and negative if all instances in the bag are negative (Babenko 2008). A Multiple Instance Learning Approach will in some problems capture some important aspects, such as it might be more realistic that at least one daily log before a failure carries a failure signature, than to assume that all the daily logs within a failure carry a failure signature. The goal is thus to correctly label a bag, rather than every instance, with the aim to tackle the problem of falsely assign unrelated events to the positive class (events that randomly appear before the failure).

To solve this problem a L1-regularized SVM optimization was used, which is suitable for MIL cases. L1-regularization is also known as Lasso Regression, which shrinks the less important feature's coefficient to zero, and thus removing some features.

The model was evaluated after its ability to label the different bags correctly, using the precision and recall curve.

- Precision is defined as: $\frac{TruePositives}{TruePositive+FalsePositive}$
  Where a true positive is a correct labeled positive bag and a false positive the contrary.

- Recall is defined as: $\frac{TruePositive}{AllFailures}$

In their research the models containing 300 to 400 predictive features achieved the best performance. Given that the requirements is 70% precision, their best model can cover 25% to 80% of failures within their predetermined predictive interval for the dataset, which was set to 7 days. Though their most complicated models contained up to 400 features, their models containing far less features also gave a high performance, with high precision and a reasonable recall score. Such models might be easier for domain experts to understand as one can easier detect the root causes of failures or identify the specific failure modes.

As seen in previous research, most machine learning methods for failure prediction form a binary classification problem, where the positive class is being generated by an interval before the failure. This predefined length of interval was in the two previous researches defined by experts. However, being able to predict the length of this interval is very essential when this knowledge is hard to acquire. In the research conducted by Korvesis, Besseau, and Vazirgiannis (2018) an interesting question is asked regarding their work on predicting maintenance in aviation; "*How many flights before the occurrence of e_τ are we able to make a prediction?*. Here e_τ is defined as a critical failure or event. By studying the event logs one can find that this failure may appear more than once, and the survival time corresponds to the time interval between two consecutive occurrences of the critical event e_τ. Time interval between starting the maintenance action until its next failure. In their research they study the use of the non-parametric Kaplan-Meier approach to study the survival function of target failures. Their object was to develop an alerting system that will notify engineers about upcoming failure, in such a manner that the engineers will be given enough time to prepare the required maintenance actions. Also in this approach a multiple instance learning scheme was applied, as seen in the other researches as well. Again, this approach is used to handle the fact that a lot of unrelated events might happen before a critical event. The target events are extremely rare, with an extremely large feature space. Hence a model that outputs a risk function based on present events was created, that quantifies the risk of an upcoming failure. Their prediction problem was classified as followed:

Let $\varepsilon = e_1, e_2, ...e_k$ be the finite set of possible events. The goal of prediction in event-logs is to predict the next occurrence of a target event $e_\tau \in \epsilon$. Partition the data set into *episodes*, corresponding to the periods between consecutive occurrence of the target event e_τ for each system. Each episode $e\rho_i^j$ of the system ρ begins with the first event after the occurrence of e_τ and end with the next occurrence of the target event.
Their goal was to learn a function $r(x^i)$ for a target event e_τ, $r : \mathbb{R}^{|\varepsilon|x1}x \mapsto [0,1]$, that quantified the risk of event e_τ occurring in the near future, given the event that occurred at time-step i, x_i. A regression problem was formulated, using the random forest regression model which gave better results than LASSO, Support Vector Regression and Gradient Boosting. In order to tackle the problem that a lot of events might not influence the critical event the researchers made some assumptions regarding removing event that appear less than n times, removing multiple occurrences of events in the same segment and penalising frequent events (because the most frequent events usually never gives any significant information). In order to decide whether a critical event e_τ will occur a simple thresholding approach was used.

The features in the model consists only of events of failures, and they identified the candidate predictors ρ^{e_τ} from the weights assigned by a random forest algorithm. The most important events was ranked by random forest. In their result the event with the highest score corresponds to a type of failure that belong to the landing gear of the airplane, which was the scope of their research. Their research (Korvesis, Besseau, and Vazirgiannis 2018) was the first attempt to perform failure prediction given only post flight reports, and the performance of their model could increase by exploiting information from a wider range of sources, such as sensors, maintenance logs and environmental variables. The three previous problems are all tackled as a binary classification problem, using a multiple-learning instance set-up. Another method was studied by K. Zhang et al. (2016) with the use of a recurrent neural network, more specific a Long Short-Term Memory (LSTM), in order to deal with the rare occurrence of the critical events. The method was applied on the event log of an IT system. The researchers state that one of the challenges with log-based failure prediction is the extremely large number of features required to achieve acceptable performance. They developed a system that was built on text mining and deep learning methods, which can be applied in hydrogenous logs, which differs from the previous methods seen in other log-based predictive maintenance research. The problem for predicting failure was more or less defined in the same manner as seen in the previous researches: A component κ and a collection of logs $L(\kappa)$ from this component, find the probability of failure $\pi_f(W)$ within the time-window W. Similar to the other approaches the problem is treated as a supervised learning problem, a binary classification problem for predicting failure events within a time window. This method differs from the other methods because the model provides the probabilities of failure, and by providing such probabilities the model can possibly be used to make safe predictions.

3.2.3 Uncertainty & Deep Learning

When creating a model, there are two main type of uncertainties one can model, respectively the aleatoric uncertainty and and the epistemic uncertainty. The former captures the noise in the observations and the ladder accounts for the uncertainty in the model it self (Jain and Srijith 2019). It has been proven to be difficult to model the epistemic uncertainty for data-driven models. In comparison Bayesian models, which is a statistical model, reasons about the model uncertainties, though at the expense of increased computational cost. Deep learning algorithms are today able to map high dimensional data to an array of outputs, but these mappings are done inside a 'black box' and assumed to be accurate. In the research by Jain and Srijith (2019), the author mentions a recent major accident where a white side of a trailer was mistaken for a bright sky in an assisted driving system, causing a fatal car crash. If such a system would have been able to assign a high level of uncertainty to their predictions, the system would have been able to make better decisions and possibly the disaster could have been avoided. As mentioned earlier, deep learning have in the recent years gained a lot of attention, specially in fields such as physics, biology and manufacturing. Then again, these are disciplines where displaying the models uncertainty is crucial. The standard tools for regression and classification do not capture model uncertainty, but in most practices and fields where deep learning tools are used a models uncertainty is indispensable. In a classification problem, for example, if the labeled output was marked with a high degree of uncertainty, one could decide to delegate this input to a human being for classification, ensuring correctness.

3.2.3.1 Uncertainty Based Models

In a research paper by Gal and Ghahramani (2016) they show that it is possible to cast deep learning tools as Bayesian models and show in their research that the use of dropout in Neural Networks can be interpreted as a Bayesian approximation to the probabilistic model Gaussian Process (Gal and Ghahramani 2016). Dropout refers to dropping units in neural networks, and has mainly been used in deep learning to avoid over-fitting, see figure 3.4. In the research done by Gal and Ghahramani (2016), a tool is developed for representing model uncertainty by extraction information that has been thrown away.

The area of active learning has been broadly studied before, but with the rapid development of deep learning, it has seen widespread interest. In 2016, Yarin Gal and Zoubin Ghahramani proposed Monte Carlo Dropout (MCD) as a method to obtain uncertainty estimates in the DL domain (Gal and Ghahramani 2016). The idea of MCD is to train a DL model to obtain an approximate posterior distribution. This is done by running multiple forward passes trough the model with a different dropout masks every time. By computing the average and the variance of sample x, an ensemble prediction is gained.

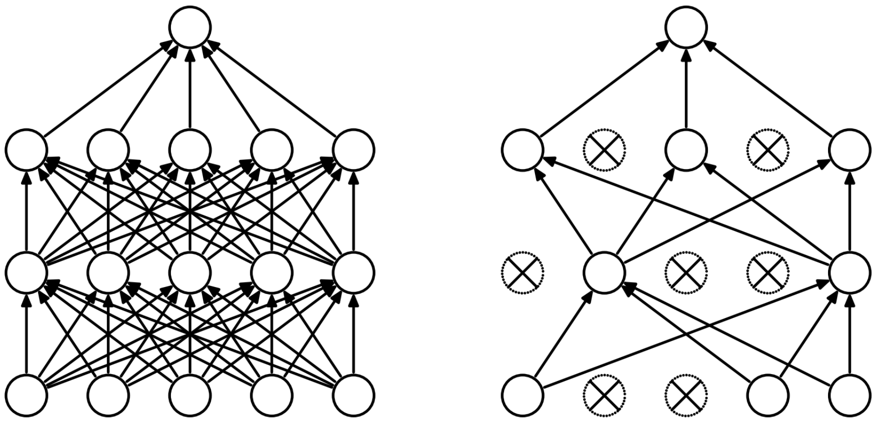


Figure 3.4: Dropout Netual Net Model

Predictive posterior mean:

$$p = \frac{1}{T}\sum_{i=0}^{T} f_{nn}^{d_i} \tag{3.3}$$

Uncertainty:

$$c = \frac{1}{T}\sum_{i=0}^{T}\left[f_{nn}^{d_i}(x) - p\right]^2 \tag{3.4}$$

To estimate the predictive mean and the predictive uncertainty stochastic forward passes is collected through the model. Furthermore, this information can be used with existing neural network models that has been trained with dropout (which, as mentioned earlier, is already widely used).

In a study done by Ramakrishnan et al. (2018), using MCD as a estimator for increasing the generalizability of a deep neural network for speech enhancement, shows that the use of MCD improves enhancement performance, especially on noise and SNR[1]. The advantage of the proposed MCD can be summarized into two main points; improved speed and greater flexibility.

Recently Nguyen and Medjaher (2019) proposed a dynamic predictive maintenance framework with the use of deep learning. In their study the whole process, from data-driven prognostics to maintenance decision is considered, and also investigating the impact of imperfect prognostics on maintenance decisions. Their framework provided the system failure probabilities in different time windows and also making an instantaneous maintenance or inventory decision based on the prognostics information. By the use of a Long Short-Term Memory network the estimated probabilities that the system will fail in different time windows in the future are given.

[1]Signal-to-noise ratio

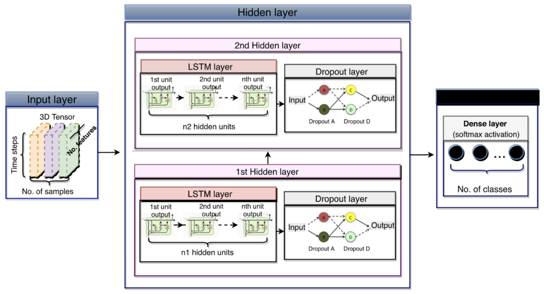


Figure 3.5: Long Short-Term Memory Network for Prognosis

These time intervals are defined according to the requirements of the operation planner. Their approach is classified as followed: Assuming N identical components monitored during their operation by m sensors. Thus, the monitoring data for each component $i = 1,2,3....N$, during the lifetime T is denoted as the matrix:

$$X_i = [x^1, x^2, ...x^t, x^{T_n}], X_i \in \mathbb{R}^{m \times T_n} \tag{3.5}$$

where $x^t = [x^t_1, x^t_2, ...x^t_m]$ is a vector of the sensor measurements. Hence, during stage one, the training stage, the LSTM network takes the sensor measurement sequences X_i to learn which time window the true RUL belongs to, and during the test stage, the LSTM classifier will take the vector measurements x^t as the input data and calculate the probability that the RUL belongs to the determined time window. In their real application case study data provided from NASA was used, with the scope to estimate which time window the systems RUL would fall into. An illustration of the output provided by the LSTM-classifier is shown in figure 3.6.

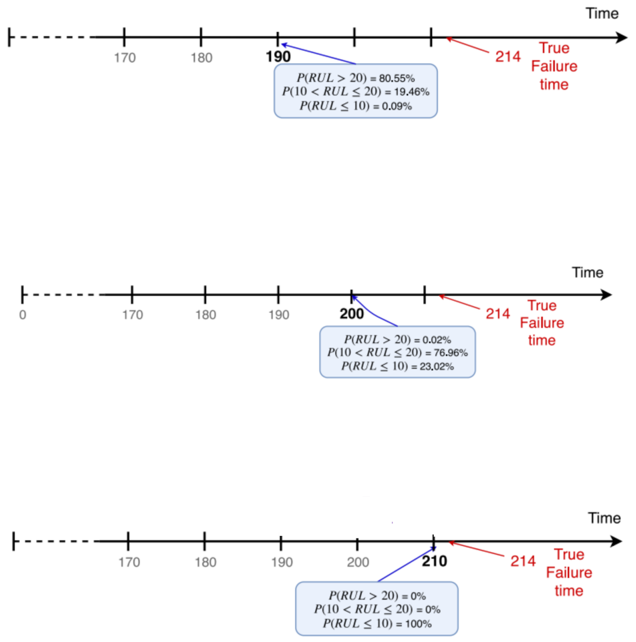


Figure 3.6: Illustration of the research conducted by Nguyen and Medjaher (2019)

In their research the time line was split into three, w_0, w_1, w_3 and a confusion probability matrix was used to evaluate the performance of the method. When the system belonged to window w_2, the mean confusion probabilities that the system belongs to w_0 are very low for all test sets, with the worst case being 2.9%. When the system is in $w_0$the mean confusion probabilities of the state being in w_0 is also non-significant. But then the system belong to w_1, the predicted state is in most cases w_2. This phenomenon can be explained by the fact that the two time windows are so close together, hence the characteristics for the different states are too similar. Thus the authors aims develop the model further in the future, to overcome such obstacles.

3.2.4 Conclusion of the Literature Review

This literature review was conducted in order to understand the state-of-the-art of machine learning approaches in predictive maintenance. The fields of anomaly detection, diagnosis and prognosis are heavily researched and applied in the industry,whereas the uncertainty aspect is less developed. This is mostly due to the current state of most machine learning methods, where uncertainty is not quantified. A machine learning model alone does not provide a sufficient uncertainty quantification, in order to be able to make safe predictions. One can argue that a prediction given without a level of uncertainty is not a valid prediction at all, hence a combination between statistical models and machine learning model is the only valid solution for prognostics to date.

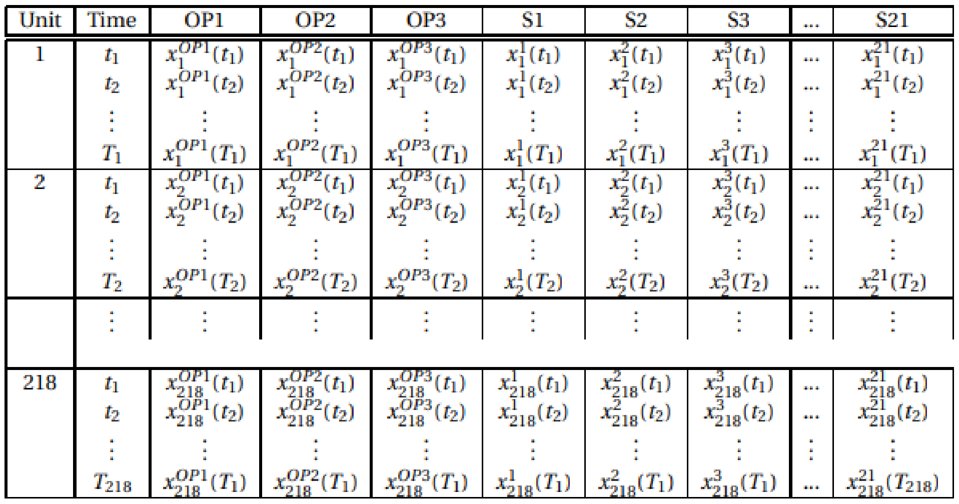
Chapter 4

Case Study

This chapter presents a case study performed in the fall semester 2019, which implements and evaluates the state-of-the-art techniques in data-driven prognostics researched in chapter 3. The data and problem description is given by the Prognostics and Health Management (PHM) society as a part of their 2008 prognostic challenge. The PHM society hosts yearly different prognostics challenges, in order to advance the data-driven prognostics discipline. The topic in the 2008 challenge was aerospace systems, where system safety is critical. A system failure can have dramatic consequences and a robust prognostic system can aid in fulfilling safety requirements, as well as reducing maintenance costs. The task was to implement a data-driven prognostic model using machine learning in order to predict the RUL of aircraft engines.

4.1 Data description

The data is provided by NASA and consists of simulated aircraft engines. It's divided into a training— and a test—set, each of which have 218 simulated aircraft engine units. Each unit is measured with the same set of sensors, where 21 of the sensor measure unit condition, and the remaining 3 sensors are concerned with the operational settings. The structure of the data is shown in 4.1.



Unit	Time	OP1	OP2	OP3	S1	S2	S3	...	S21
1	t_1	$x_1^{OP1}(t_1)$	$x_1^{OP2}(t_1)$	$x_1^{OP3}(t_1)$	$x_1^1(t_1)$	$x_1^2(t_1)$	$x_1^3(t_1)$	...	$x_1^{21}(t_1)$
	t_2	$x_1^{OP1}(t_2)$	$x_1^{OP2}(t_2)$	$x_1^{OP3}(t_2)$	$x_1^1(t_2)$	$x_1^2(t_2)$	$x_1^3(t_2)$	...	$x_1^{21}(t_2)$
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	T_1	$x_1^{OP1}(T_1)$	$x_1^{OP2}(T_1)$	$x_1^{OP3}(T_1)$	$x_1^1(T_1)$	$x_1^2(T_1)$	$x_1^3(T_1)$	...	$x_1^{21}(T_1)$
2	t_1	$x_2^{OP1}(t_1)$	$x_2^{OP2}(t_1)$	$x_2^{OP3}(t_1)$	$x_2^1(t_1)$	$x_2^2(t_1)$	$x_2^3(t_1)$	...	$x_2^{21}(t_1)$
	t_2	$x_2^{OP1}(t_2)$	$x_2^{OP2}(t_2)$	$x_2^{OP3}(t_2)$	$x_2^1(t_2)$	$x_2^2(t_2)$	$x_2^3(t_2)$	...	$x_2^{21}(t_2)$
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	T_2	$x_2^{OP1}(T_2)$	$x_2^{OP2}(T_2)$	$x_2^{OP3}(T_2)$	$x_2^1(T_2)$	$x_2^2(T_2)$	$x_2^3(T_2)$	...	$x_2^{21}(T_2)$
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
218	t_1	$x_{218}^{OP1}(t_1)$	$x_{218}^{OP2}(t_1)$	$x_{218}^{OP3}(t_1)$	$x_{218}^1(t_1)$	$x_{218}^2(t_1)$	$x_{218}^3(t_1)$	...	$x_{218}^{21}(t_1)$
	t_2	$x_{218}^{OP1}(t_2)$	$x_{218}^{OP2}(t_2)$	$x_{218}^{OP3}(t_2)$	$x_{218}^1(t_2)$	$x_{218}^2(t_2)$	$x_{218}^3(t_2)$	...	$x_{218}^{21}(t_2)$
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	T_{218}	$x_{218}^{OP1}(T_1)$	$x_{218}^{OP2}(T_1)$	$x_{218}^{OP3}(T_1)$	$x_{218}^1(T_1)$	$x_{218}^2(T_1)$	$x_{218}^3(T_1)$	...	$x_{218}^{21}(T_{218})$

Figure 4.1: Illustration of the PHM2008 dataset (Barros 2019)

Table 4.1 shows some statistics regarding the dataset. The shortest of the recorded engines, consists of 127 cycles, while the largest consists of 356 cycles. It is assumed that the last recorded sample for each unit in the training set is considered a failure, whereas in the testing set, the simulation stops some time prior to failure. Each simulated unit begins with a different level of degradation, but each initial unit condition are considered normal.

Statistic	Run length
Minimum	127
Maximum	356
Average	209
Standard deviation	43.5

Figure 4.2 shows the relationship between the three sensors regarding operational settings. Their values cluster around six ranges of values, meaning that the system experiences six different operational regimes. Data inspection and plotting shows that all unit condition sensors ranges between six cluster corresponding to their respective operational regime. An example is presented in figure 4.3 (b) showing the correlation between unit-condition-sensor 1 and 2 for the first unit in the training set. Sensor behaviour is illustrated in figure 4.3 (a) where the values of one sensor throughout a simulation is plotted. The sensor values are rapidly changing making it more difficult to capture the trend of the data. It's now known that the data was designed to confuse, and not fully represent normal aircraft engine behaviour.

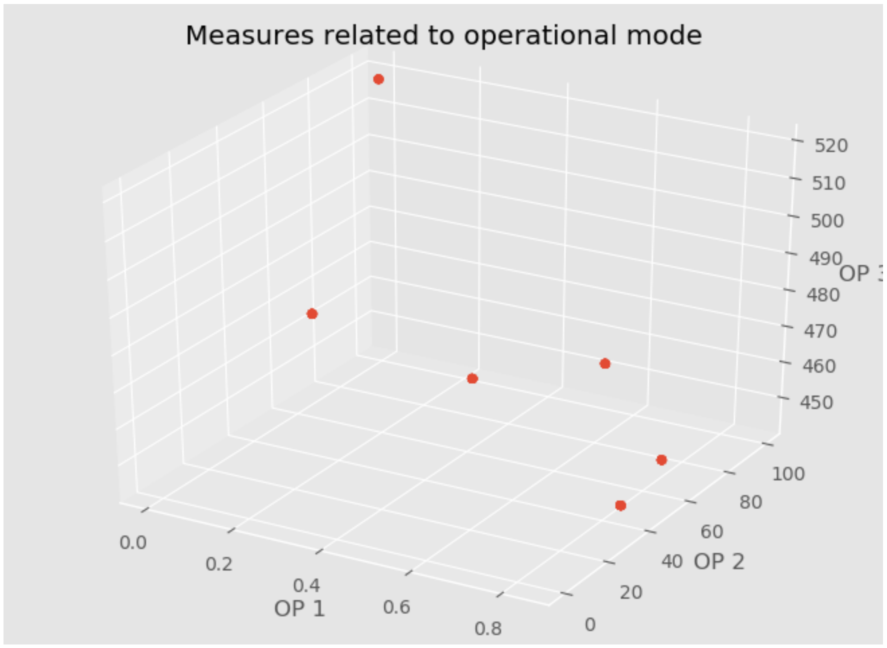


Figure 4.2: Relationship between the three operational modes

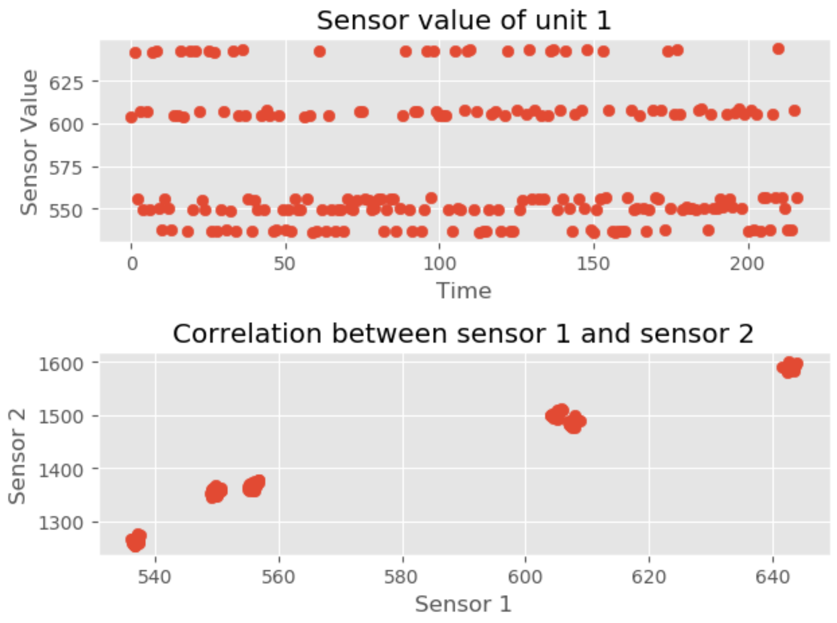


Figure 4.3: Figure (a) shows the behaviour of a unit condition sensor throughout a simulation. Figure (b) shows that correlation between two unit condition sensors

4.2 Data preprocessing

4.2.1 Standardization & Normalization

Feature scaling is an important preprocessing step in machine learning. Many machine learning algorithms use some sort of an optimization algorithm in order to adjust the feature weights. Such optimization algorithms might be gradient descent or the Levenberg-Marquardt algorithm (Moré 1978). Originally, the different features in the dataset often range between different values and have different units. For instance, there could be a dataset with a feature measuring temperature, typically between -30 to 40 degrees, and then have another feature measuring pressure within the range of 500 to 2000 Pa. The difference in the range of values of these features is large, and without any form of scaling, it might result in certain feature weights updating faster than others. This will then lead to a lower prediction accuracy (Juszczak, Tax, and Duin 2002). There is mainly two ways to scale the data:

- Standardization, is when the features are scaled in a way that they have the properties of a standard normal distribution with mean value = 0 and standard deviation = 1. When performing this standardization, the creator is assuming that the dataset fits the Gaussion distribution with a well behaved mean and standard deviation. It is possible to standardize a dataset that doesn't have this assumption, but the result might not be reliable.
- Normalization, is when the features are scaled to a fixed range, often [0,1] or [-1,1].

Due to the heterogeneous data, normalization has been performed on the dataset, ranging it's values from 0 to 1. Many machine learning approaches, and especially neural networks are sensitive to the input data range, making the scaling step a necessity (M.-L. Zhang and Z.-H. Zhou 2004).

4.2.2 Principal components analysis

Principle component analysis (PCA) a statistical procedure that locates correlated features from the dataset and converts it into uncorrelated features (Wold, Esbensen, and Geladi 1987). By example, in the PHM2008 challenge dataset, there might be features and/or combination of features that display the same information. Though it isn't a challenge in this dataset, having huge amounts of data with a great number of features that needs to be processed in real time can make a problem too computationally complex. PCA will then find the redundant information, and reduce the number of features in order to reduce the complexity, as well as maintaining most of the information.

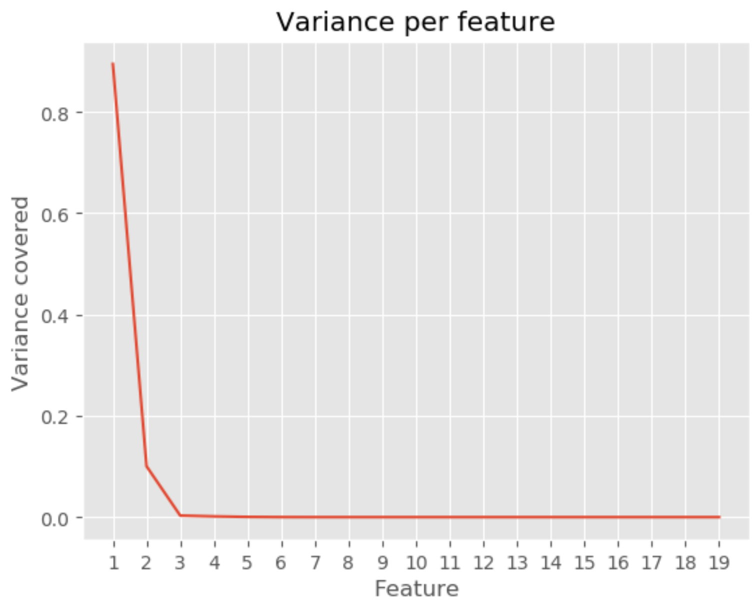


Figure 4.4: Graph showing the covered variance from each of the 19 chosen features after PCA is run.

A PCA algorithm from the sklearn[1] library is performed on the dataset, where the algorithm suggests to reduce the number of features down to 19. Figure 4.4 shows the percentage of variance covered by each of the selected features. By inspecting figure 4.4, it shows that using two features covers most of the variance in the dataset. That said, due to the confusing behaviour of the simulation, all features are used to capture the trend in terms of degradation.

4.3 Methodology & architecture

4.3.1 Problem strategy - Regression vs classification

This problem can be viewed as a regression problem or a classification problem. A regression problem approximates a mapping function from the inputs to a continuous output, a RUL estimation in this case. A classification problem approximates a mapping function from the inputs to a discrete output (chapter 2.3.3.1). In this case, it could be a binary classification problem by assigning a new unseen unit to a class dependent on whether it will fail or not within the next t time units. Both strategies has been implemented.

[1]https://scikit-learn.org/stable/

4.3.2 Model choice

Similar to other case studies presented in chapter 3, the data is both multivariate and sequential. Recurrent neural networks has shown great performance on similar problems with similar data. Three types of recurrent neural networks are implemented for comparison, a Simple Recurrent Neural Network (SRNN), a Long-Short Term Memory network (LSTM) and a Gated Recurrent Unit network (GRU). In contrast to the traditional acyclic feed forward neural network, which assumes that all inputs are independent (chapter 2.3.5), the RNN makes use of the sequential information that the input may have (Mitchell 1997). This property is important when predicting a future value based on the trend of the historical data. Out of the three networks, the SRNN is the one that incorporates sequential information the least and it is expected to perform worse than the LSTM and GRU (Schmidhuber, F. Gers, and Eck 2002).

4.3.2.1 Evaluation metrics

The evaluation metric is a measure of the algorithm's performance. The implemented models are supervised learning methods which means that the true targets are known. The algorithm's performance can then be evaluated by calculating the distance between the predicted value and the true value. There exist a range of different methods that calculates this distance, and in this particular case, the Mean-Squared Error (MSE) is used. Given vectors of predictions ($\hat{Y}$) and true values (Y), then:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2, \tag{4.1}$$

where n is the number of predicted samples.

4.3.2.2 Model configuration

- **Error metric**: Mean-Squared Error
- **Number of epochs**: The number of epochs refers to how many times the model has trained on the training set. Many epoch values are experimented with, where at least 40 epochs are necessary for some loss convergence. The models are run with epoch numbers up to 60 resulting in better predictions, but not perfect convergence on the validation set. It has not been experimented with a number of epochs greater than 60 due to the limited computational resources.
- **Batch size**: The entire dataset is not computed in the network at the same time. It is partitioned into batches of 200 samples. This means that after every 200th sample a back-propagation is performed and feature weights are updated.

- **Optimizer**: The ADAgrad optimizer is used for the regression problem, while the ADAM optimizer is used for the classification problem.
- **Activation function**: The ReLU activation function is used for the regression problem and the sigmoid activation function is used for the classification problem.
- **Dropout**: A technique that randomly chooses and deactivates a given percentage of the nodes in the hidden layers. It is commonly used to reduce model overfitting (Srivastava et al. 2014), and as described in chapter 4.3.3, it can be used to implement Monte Carlo dropout.
- **Validation set**: 20% of the training set is split into a validation set.

4.3.3 Uncertainty assessment

The uncertainty is evaluated by using the Monte Carlo droupout technique proposed by (Gal and Ghahramani 2016). Uncertainty in deep learning can be modeled by running multiple forward passes with different sets of deactivated nodes in both the training and testing phase. Given a a neural network with droupout, fnn, the uncertainty of a new unseen sample x is derived by performing T predictions with different droupout masks, d_i. The model then ouputs a vector of T predictions,

$$f_{nn}^{d_0}(x), ..., f_{nn}^{d_T}(x) \tag{4.2}$$

The model posterior distribution from sample x can then be derived by calculating the prediction mean $\hat{x}$ and variance σ^2.

$$\hat{x} = \frac{1}{T}\sum_{i=0}^{T} f_{nn}^{d_i}(x) \tag{4.3}$$

$$\sigma^2 = \sum_{i=0}^{T}(f_{nn}^{d_i}(x) - \hat{x})^2 \tag{4.4}$$

4.3.4 Model evaluation

The strategy regarding model evaluation is different for the regression view and the binary classification view. For the regression problem, the total loss per epoch for both training and validation sets are the metrics that the model tries to reduce by updating the feature weights. During the iterations, the models save the weights that resulted in the lowest validation loss. Further, after iterating through all epochs, the model performs predictions on the test set with the saved set of weights. The total loss on the test set is the final evaluation of the model.

With regards to the classification view, the accuracy (percentage of correctly classified samples) is the metric of interest. It follows the same procedure as with the regression problem, where the feature weights resulting in the highest accuracy score on the validation set is saved. These weights are then selected when performing classification predictions on the test set.

Chapter 5

Results & Analysis

5.1 Model results

The task was first considered as a classification problem since it required less computation power in order for the models to perform acceptably. During the classification problem, the models converged after approximately 15 epochs, while the regression problem required around 60 epochs before convergence was reached.

The target value in the training set was changed from remaining time to failure to a binary value, either 0 or 1. A target value of 0 meant that the particular unit would not fail within a predefined period, while 1 represented a failure within that period. The resulting models could then output a probability for an unseen unit failure within the given period. The models were training on different lengths of the predefined period. Below shows the result for the three models (SRNN, LSTM, GRU) with a period = 30 time units.

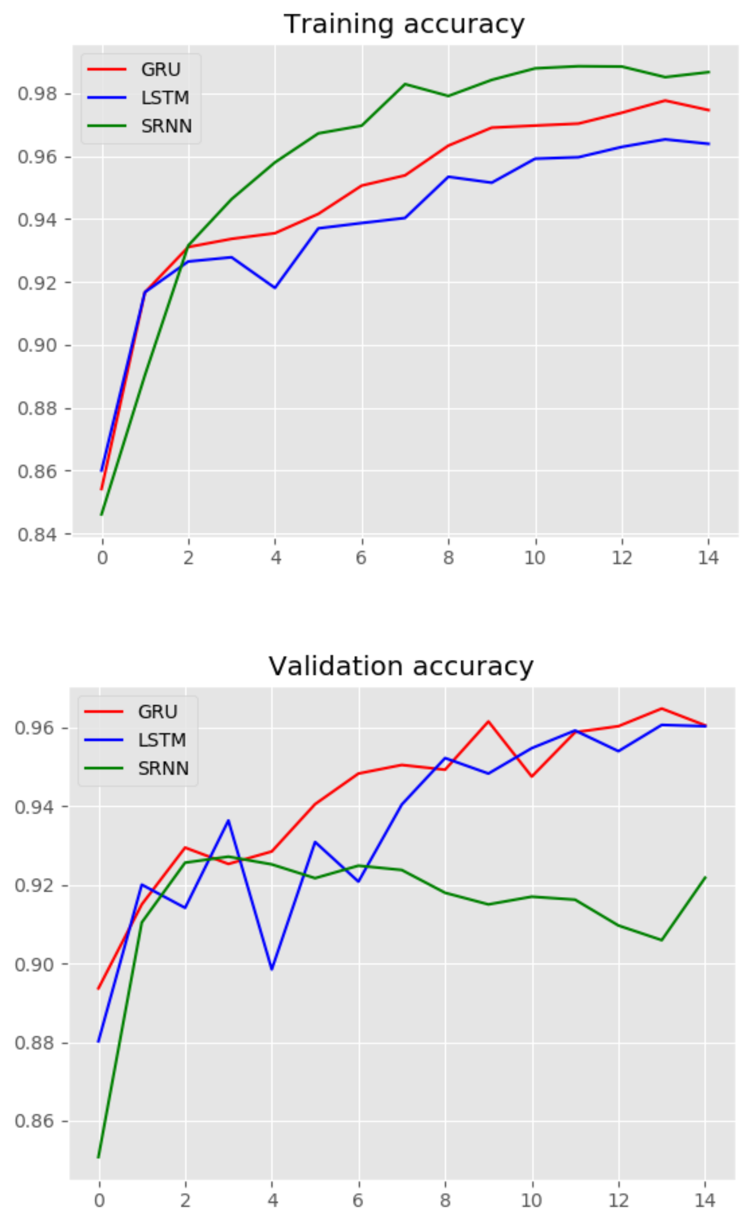


Figure 5.1: SRNN, LSTM and GRU model accuracy. The top plot shows the training accuracy, and the bottom plot shows the validation accuracy.

The top model in figure 5.1 shows the accuracy for the training set per epoch. All of the models reach convergence, where the SRNN is performing the best, GRU is second and LSTM last. Though, looking at the accuracy for the validation set in the bottom plot, shows that the SRNN performs far less than the GRU and LSTM. This is typical traits of overfitting, where the SRNN is performing well on the training data, but doesn't generalize well enough to accurately predict unseen units. In addition, the GRU performs better than the LSTM on both the training— and validation— set. Final accuracy scores for each model is presented in the table below.

Model	Training Accuracy	Validation Accuracy
SRNN	0.977527	0.960272
LSTM	0.953545	0.976082
GRU	0.973829	0.978751

5.2 RUL estimate & model uncertainty

The GRU model performed best in the classification problem and is therefore the chosen model in the regression problem. In order for the model to convergence it required a lot more epochs iteration than in the classification problem. Hence, it required more computational power, and due to the limiting hardware used, only one of the models were trained. After 60 epoch iterations, the model had reached some convergence, though the loss were still quite high. A lot of parameter tuning were performed in order to reduce the loss with little luck. It's known from the provider of the dataset that it was designed to confuse the model. Further research found examples of other structure strategies of the data that resulted in some better predictions. This is not implemented in this model, since the main objective was to evaluate different models in RUL estimation, and particularly strategies to model their uncertainties.

After 60 epochs iterations, the GRU model was run on 100 unseen units. Figure 5.2 shows the the predicted RUL values (red line) and the actual remaining life of the 100 units, while figure 5.3 also display the uncertainty of the prediction in terms of two standard deviations.

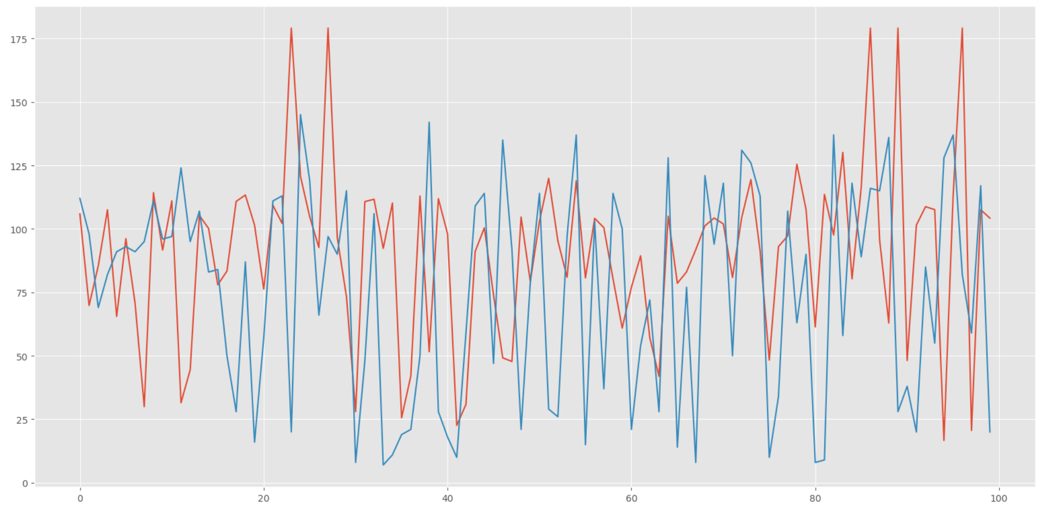


Figure 5.2: RUL prediction on 100 unseen aircraft engine units

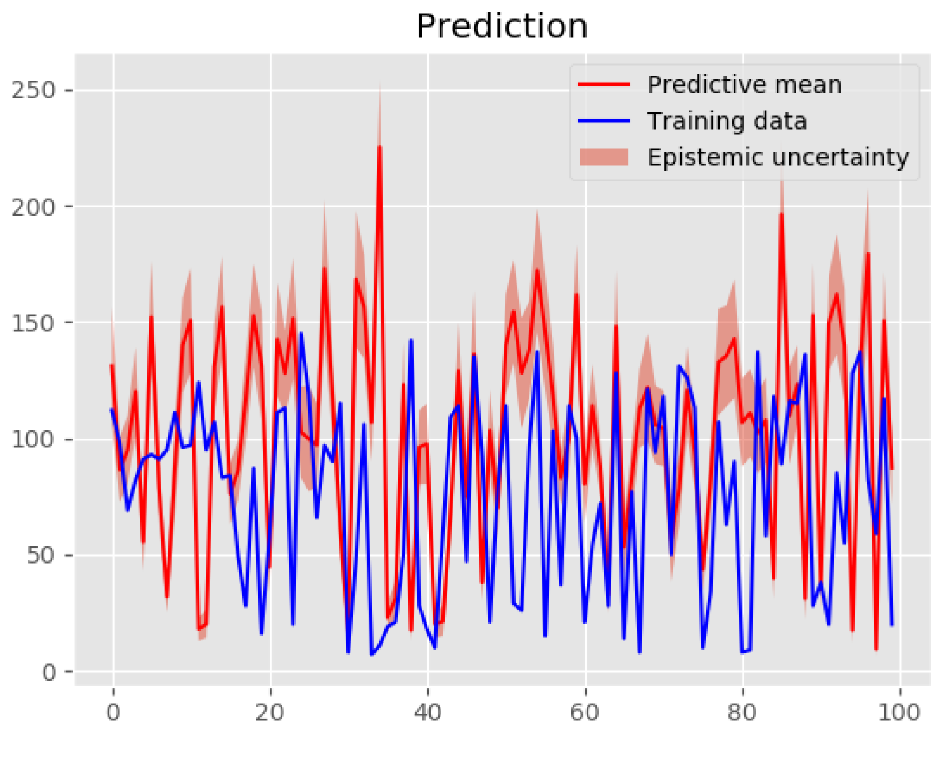


Figure 5.3: RUL prediction on 100 unseen aircraft engine unit with uncertainty

Chapter 6

Discussion

6.1 Literature review

As a remainder the research questions for the literature review was:

- **Q1**: What machine learning approaches are most commonly used in PdM?
- **Q2**: To which and what extent has ML-approaches been used in anomaly detection, diagnostic and prognostics, respectively?
- **Q3**: Which fields are the less researched, and what are the challenges?
- **Q4**: To what extent can ML-approaches be used as a solid ground for decision making regarding maintenance strategies and provide a good uncertainty quantification?

6.1.1 Q1

The literature shows that the use of data-driven machine learning approaches in predictive maintenance is of great interest due to its capabilities of capturing patterns in complex systems. Chapter 3 displays multiple condition monitoring and predictive maintenance system with machine learning. There is a combination of both traditional machine learning approaches, like SVM, KNN, and different regression techniques, and different versions of neural networks. Condition monitoring systems often consists of a large number of sensors resulting in lots of generated data. Artificial neural networks and deep learning have in several domains outperformed both model-based approaches, as well as traditional machine learning approaches. Hence, its not surprising that many of the new condition monitoring systems tries to implement some sort of ANN.

6.1.2 Q2

6.1.2.1 Anomaly Detection

The use of machine learning in anomaly detection has greatly increased the last two decades. Recognizing patterns and abnormal behaviour in complex systems is computationally challenging, where the use of machine learning and deep learning has shown great potential. That being said, there exist challenges in anomaly detection that cannot be solved by only using machine learning, but require traditional methods and domain knowledge. In order for a model to separate normal and abnormal behaviour, a clear understanding of normal behaviour is required. There exist mainly two challenges in determining normal behaviour, (i) training the model with data that is not contaminated, and (ii) there might exist multiple normal behaviours dependent on the operational regimes the system experiences. Data without contamination is rare in real life systems, and some filtering techniques are required. The applications presented in chapter3.2.1 are dependent on some suitable mathematical function as a preproccesing step on the data, before its fed into a machine learning model. The second challenge has to some extent been solved using unsupervised data mining techniques, like clustering algorithms, though it is still required to implement some domain knowledge. Therefore, determining the threshold for normal behaviour remains a difficult task for machine learning alone. What has been observed is that as condition monitoring systems become more advanced, traditional anomaly detection systems are not sufficient enough. Some sort of machine learning technique is required, and can enhance the performance in combination with domain knowledge.

6.1.2.2 Diagnostics & Prognostics

Multiple data-driven prognostics models have been applied to condition monitoring systems, mostly in form of RUL estimations. Due to IoT, condition monitoring systems contain a lot of heterogeneous data that are constantly changing and arrives at different speed, which makes machine learning a suitable tool and a necessity. Many of the most developed data-driven condition monitoring systems are applied in rotating machinery. Rotating components exist in many industries, and often based on measured vibration. The applications reviewed in chapter 3.2.2.2 are mostly concerned with creating high accuracy ML models with precise RUL predictions. Obtaining knowledge of which features that have the most impact is therefore important, though information of which features that might contain faults are rarely highlighted in the different case studies, The diagnostic part is a necessity in order to perform predictions, but it seems like the main goal of the models are to get a high accuracy on the test set, and not identify what causes a potential failure. Many of the applications become a model-parameter tuning task, rather than cause-failure identification system. This might be due to the "black-box" property of ML models, where its difficult to extract information and follow the model behaviour through training.

Similar to anomaly detection system, there is a strong desire to implement ML models and especially deep learning approaches, due to its capability to handle complex systems. That being said, many of the authors emphasise that combinations of ML models are a necessity, due to one model on its own not being able to capture all possible scenarios. Combination of models are used either as a data preprocessing step or as an ensemble of models (chapter 3.2.2.3).

6.1.3 Q3

Of the reviewed applications from the literature, few have implemented techniques trying to describe the uncertainty of the predictions. This was expected, and is discussed in literature question 4 below. In addition, separating the diagnostic part from the prognostic part is not straightforward. Most of the reviewed studies in chapter 3, implement a combination of both, with emphasize on the final RUL estimation. There is little too none discussion on what part of the system or which features that have the most impact on system degradation. A robust predictive maintenance models must be able to locate potential failure causes in order to perform optimal maintenance.

Another disadvantage is the requirement of having comprehensive data such that the machine learning model will perform well. As explored in the case study, in order to converge properly and generalize well on new data, it is required that the training data sufficiently represents the targeted domain. Especially, when predicting remaining useful life or time to failure, it is required that the model has trained on measured systems until failure. This is not always the case in real life applications, where it can be costly or even impossible to monitor systems until failure.

6.1.4 Q4

Modeling uncertainty with ML models is a challenging task but necessary in order to create a trustworthy prediction model. Simply entering raw data into a model and calculate an estimate of remaining life without any information regarding prediction confidence is not sufficient enough. Most of the application reviewed in chapter 3 does not provide this information. In recent years, some researchers have proposed techniques to overcome this challenge. Probabilistic neural networks, like Bayesian neural network with a Monte Carlo droupout technique has made it possible to retrieve a posterior prediction distribution for new samples. As explained in section 2.2, randomly deactivating a percentage of the nodes in the hidden layers for each iteration, results in multiple RUL estimations for a new sample. From the array of estimations, the mean and variance can be calculated, and fitted in a probability density function. Each new sample are now not only given a RUL prediction, but also a display of the model's uncertainty for that particular sample. The use of Monte Carlo droupout has not only provided uncertainty information, but also enhanced the performance of the algorithm. The field of uncertainty within machine learning remains limited, but has in recent years become a research field of high interest.

6.2 Case-study

A case study was completed in order to test and evaluate different techniques reviewed in the literature. Data was required, and the PHM challenge looked like a suitable approach. As mentioned in chapter 4, the task was to calculate the remaining life of aircraft engines. There was no emphasis on displaying the confidence level of the algorithm nor explain the reasons for the estimations. This might be reflected in the sense that the challenge was from 2008, and it's not before in recent years that the topic of ML model's uncertainty has been discussed properly. There was more focus on high accuracy models with complex parameter tuning techniques, rather than implementing a robust prediction model. The data was designed to confuse the algorithm which made it challenging to predict good estimations. That being said, tuning a model on contaminated data is good practice for real life applications. The resulting case study implemented multiple deep learning methods, evaluated their performances and to some extent described the confidence level of the predictions. Although, the collected results weren't particularly impressive, valuable experience with the implementation of data-driven prognostic models was obtained.

Chapter 7

Future Work - Collaboration with Teekay

7.1 Introduction

In our master thesis we are going to collaborate with the company Teekay. After several meetings throughout the semester, the agreement between us and the company resulted in us writing about the company and the information systems they use in order to get a deeper understanding. In this way, we will be better prepared for the upcoming Master Thesis. Therefore, we will give an introduction to this as a final chapter.

7.2 Teekay

Teekay is a company working in the field of providing floating oil platforms called FPSO[1] to oil and gas companies. Teekay's core values are safety and sustainability - with safety first. Hence, we want to investigate the opportunity for using their available data in order to contribute to increase the safety in their working environment even more. Teekay generate vast amounts of metadata, which is all used for different purposes. Then again, the company lacks a streamlined collaboration between their integrated information systems in order to be able to expose the full potential in their generated data.

7.3 Information Systems

To prepare for the future collaboration with Teekay the need for a clear understanding of their information systems is essential. Therefore, in this section a brief introduction to their three main information systems is presented; PI, Star IPS and IP21 respectively.

[1]Floating Production, Storage and Offloading

7.3.1 PI

The PI-information system is a software by OSIsoft, which collects data from several data sources with the aim to streamline processes and drive continuous improvements. First order of business is to capture the data, search and analyse, third is to visualize and the final step is to share trends and discoveries from the data in order to create value for the business. The information system is hence capable of giving a historical, real-time and predictive insight to the ongoing processes in the business. An important aspect of this information system is the connection capabilities, because the system depends on the collection of time-based data from a diversity of systems. This great diversity allows the system to include data from various sources, such as SCADA, DCS, equipments, databases, text files and HTML pages. The main and most important component of the PI system is the PI server, which consist of different technologies in order to optimize data storage, transformation and delivery. This server includes tools used for calculation to transform raw data, combine data values from multiple systems into analyses, calculations and so on. As always, the main challenge is assigning the right data to the right tool in order to get valuable insight to the process of the business. The PI system provides a wide range of such tools that take the complexity of out data to access, visualization and analysis.

7.3.2 Star Information & Planning System

Star Information & Planning System is an easy-to-use business application software system which enables rig and ship owners to operate the fleet in a safe and efficient way. It consists of different integrated modules which can be combined to meet a variety of user demands. The system is designed for handling several tasks within ship management which can consist of both simple maintenance and spare part handling, as well as more complex, comprehensive information sharing and strategic planning. Operational deviations and experiences as well as corrective measures can be logged and reported in the system.

The Star IPS will contain one vessel which will always be in focus. Normally the system is used in interaction with office based systems such as Star Fleet Management System[2]. However, the system may also be used as a stand-alone vessel system without shore based interaction.

7.3.3 IP21

The Aspen IP.21, Info-Plus 21, is known to be a powerful tool used in a variety of fields of oil and gas production. The IP.21 is a data historian which enables to access pilot plant process data, collected from the Process Control Systems, Distributed Control System or equivalent systems. The system is validated for collection real time GMP data, Good Manufacturing Practice

[2]Star Fleet Management System is a system for fleet, purchase, QA and insurance managers.

for ensuring that components are controlled according to quality standards. The system is used to improve work practices and reduce downtime, using real-time data from multiple sources to create a complete overview of the operations. IP21 can be used for effective preventive maintenance by avoiding unexpected downtime, and provides improved process analysis and time saving as the system automatically gathers gate in order to give an overview of the whole system and the state of the different components.

7.4 Collaboration with Teekay

Knowing that Teekay has a lot of different information systems, the scope of our future work is to verify if the faults detected during the routine control could have been predicted before hand, so downtime could have been avoided. The company has a lot of system logs which is written after such routine controls which states if the valve is in a faulty state or not and if a failure has occurred. Thus the first order of business is to create a machine learning model using text recognition to gather relevant data from these system logs. To our knowledge, these system logs are written on a standard format, so we aim to extract the time and date for the failure and the maintenance done on the valve. Being able to extract such information about the failure time for the valve, we wish to integrate this information with the measurements for the valve which is gathered in a different information system. Thus, we can integrate the different data sources, we will have a data set consisting of different measures (pressure, temperature e.g.) and also the failure time from the valve. Hence, we want to investigate the correlation between the failure times and the measured values before hand of such an event and potentially create a machine-learning based approach for being able to make prediction about the remaining useful time of the valve. Teekay is a company operating in the oil and gas industry, and thus safety is one of their main concerns. Hence, the importance of making safe predictions is crucial. As studied in the literature review, this is still a field in machine-learning where a lot of research remains and we aim to see if we can be able to create a safe prediction models by combining machine-learning models with traditionally statistical approaches.

Chapter 8

Conclusion

The use of machine learning in condition-based monitoring and health prognostics is a field of study with great potential. The increasing amount of data allows the possibility of optimizing maintenance with more precise predictive abilities. Better and more insightful information is obtained with the implementation of sensors and several data manipulation techniques. Fast growing heterogeneous data requires efficient data processing techniques, and machine learning, as a possible solution, has been explored in this thesis.

A comprehensive literature review has been conducted, reviewing the most developed applications of machine learning in predictive maintenance. The most obvious benefit of machine learning is it's capabilities to recognize patterns in complex systems. It has shown great performance with high accuracy prediction concerning system diagnostics, evaluating the current system condition, as well as predicting remaining useful life. However, there exist some challenges and requirements when using the strong benefits of machine learning.

A well performing machine learning model requires comprehensive data. As explored in the case study, in order to converge properly and generalize well on new data, it is required that the training data sufficiently represents the targeted domain. Especially, when predicting remaining useful life or time to failure, it is required that the model has trained on measured systems until failure. This is not always the case in real life applications, where it can be costly or even impossible to monitor systems until failure. Another challenge is the trustworthiness and uncertainty aspect of the machine learning model. Its capabilities of recognizing complex patterns comes with a great disadvantage, namely explaining the causes and confidence of its predictions. This field of study, particularly in neural networks, has in the few recent years risen in popularity. Probabilistic techniques like Monte Carlo dropout has been experimented on neural networks, in order to capture the different types of uncertainty of the models.

The conducted study case displays the strong capabilities of recurrent neural networks on advanced and confusing data. Three types of recurrent networks were implemented, where the LSTM and GRU networks had the highest accuracies. With advanced parameter tuning techniques, the models generalized well with an accuracy score of 0.976 and 0.978 after just a few iterations. Model uncertainty has to some extent been extracted by the use of Monte Carlo droupout technique during the training and testing phase.

The literature review and the study case prove the great potential of machine learning models in predictive maintenance, however they are not fully capable of being a robust prediction model on their own. Though, it has proven to be a beneficial supplementary tool in combination with traditional statistical decision strategies. With increasing interest within the field of artificial intelligence, there might be some standalone machine learning prediction models in the future.

Appendix A

Case Study; Code

```
import matplotlib
import pandas as pd
import numpy as np
import time as time
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import confusion_matrix,accuracy_score

from keras.optimizers import RMSprop
from keras.models import Sequential, load_model, Input, Model
from keras.layers import Dense, LSTM, Activation, GRU, SimpleRNN #Dropout
from keras.callbacks import EarlyStopping, Callback, TensorBoard
from additional_classes import Dropout
import tqdm

import matplotlib.pyplot as plt
plt.style.use('ggplot')
#%matplotlib inline

Name = "LSTM-all-params-batch{}".format(int(time.time()))
Name2 = "LSTM-all-params-epochs{}".format(int(time.time()))

tensorboard = TensorBoard(log_dir='logs/{}'.format(Name), update_freq='batch',
    histogram_freq=10, write_graph=True)
tensorboard2 = TensorBoard(log_dir='logs/{}'.format(Name2), update_freq='epoch',
    histogram_freq=10, write_graph=True)
```

```
dataset_train=pd.read_csv('train.txt',sep=' ',header=None).drop([26,27],axis=1)
col_names = ['id','cycle','setting1','setting2','setting3','s1',
    's2','s3','s4','s5','s6','s7','s8','s9','s10','s11','s12',
    's13','s14','s15','s16','s17','s18','s19','s20','s21']
dataset_train.columns=col_names

dataset_test=pd.read_csv('test.txt',sep=' ',header=None).drop([26,27],axis=1)
dataset_test.columns=col_names

pm_truth=pd.read_csv('PM_truth.txt',sep=' ',header=None).drop([1],axis=1)
pm_truth.columns=['more']
pm_truth['id']=pm_truth.index+1

# generate column max for test data
rul = pd.DataFrame(dataset_test.groupby('id')['cycle'].max()).reset_index()
rul.columns = ['id', 'max']

# run to failure
pm_truth['rtf']=pm_truth['more']+rul['max']
pm_truth.drop('more', axis=1, inplace=True)
dataset_test=dataset_test.merge(pm_truth,on=['id'],how='left')
dataset_test['ttf']=dataset_test['rtf'] - dataset_test['cycle']
dataset_test.drop('rtf', axis=1, inplace=True)
dataset_train['ttf'] = dataset_train.groupby(['id'])['cycle'].
    transform(max)-dataset_train['cycle']

df_train=dataset_train.copy()
df_test=dataset_test.copy()

period=30
#df_train['label_bc'] = df_train['ttf'].apply(lambda x: 1 if x <= period else 0)
#df_test['label_bc'] = df_test['ttf'].apply(lambda x: 1 if x <= period else 0)

df_train['label_bc'] = df_train['ttf']
df_test['label_bc'] = df_test['ttf']

features_col_name=['setting1', 'setting2', 'setting3', 's1', 's2', 's3', 's4',
    's5', 's6', 's7', 's8', 's9', 's10', 's11', 's12', 's13', 's14', 's15',
    's16', 's17', 's18', 's19', 's20', 's21']
target_col_name='label_bc'

#--- Normalizaing the data ---------

sc=MinMaxScaler()
df_train[features_col_name]=sc.fit_transform(df_train[features_col_name])
df_test[features_col_name]=sc.transform(df_test[features_col_name])

seq_length=50
seq_cols=features_col_name

#--- Organizing the data to fit into the recurrent neural network-----

def gen_sequence(id_df, seq_length, seq_cols):
    df_zeros=pd.DataFrame(np.zeros((seq_length-1,id_df.shape[1])),
        columns=id_df.columns)
    id_df=df_zeros.append(id_df,ignore_index=True)
    data_array = id_df[seq_cols].values
    num_elements = data_array.shape[0]
    lstm_array=[]
    for start, stop in zip(range(0, num_elements-seq_length),
        range(seq_length, num_elements)):
        lstm_array.append(data_array[start:stop, :])
    return np.array(lstm_array)

# function to generate labels
def gen_label(id_df, seq_length, seq_cols,label):
    df_zeros=pd.DataFrame(np.zeros((seq_length-1,id_df.shape[1])),
        columns=id_df.columns)
    id_df=df_zeros.append(id_df,ignore_index=True)
    data_array = id_df[seq_cols].values
    num_elements = data_array.shape[0]
    y_label=[]
    for start, stop in zip(range(0, num_elements-seq_length),
        range(seq_length, num_elements)):
```

```
        y_label.append(id_df[label][stop])
    return np.array(y_label)

X_train=np.concatenate(list(list(gen_sequence(df_train[df_train['id']==id],
    seq_length, seq_cols)) for id in df_train['id'].unique()))
y_train=np.concatenate(list(list(gen_label(df_train[df_train['id']==id],
    50, seq_cols,'label_bc')) for id in df_train['id'].unique()))
X_test=np.concatenate(list(list(gen_sequence(df_test[df_test['id']==id],
    seq_length, seq_cols)) for id in df_test['id'].unique()))
y_test=np.concatenate(list(list(gen_label(df_test[df_test['id']==id],
    50, seq_cols,'label_bc')) for id in df_test['id'].unique()))

nb_features =X_train.shape[2]
timestamp=seq_length

#-------------------------------------------

#Calculating the probability of component failure within 30 days


def prob_failure(machine_id, model):
    model_pred = model
    machine_df=df_test[df_test.id==machine_id]
    machine_test=gen_sequence(machine_df,seq_length,seq_cols)
    m_pred=model_pred.predict(machine_test)
    prob_failure = m_pred
    prob_failure=list(m_pred[-1]*100)[0]
    return prob_failure

# Predicting RUL of unseen units

def predict_RULs(numOfUnits, model):
    mc_predictions = np.zeros((numOfUnits, 50))
    for j in range(0,numOfUnits):
        machine_df=df_test[df_test.id==j + 1]
        machine_test=gen_sequence(machine_df,seq_length,seq_cols)
        for i in tqdm.tqdm(range(0,50)):
            y_p = model.predict(machine_test, batch_size=200)
            mc_predictions[j, i] = y_p[-1]
    pred_mean = np.zeros((numOfUnits))
    pred_std = np.zeros((numOfUnits))
    for k in range(0,numOfUnits):
        pred_mean[k] = np.mean(mc_predictions[k,:])
        pred_std[k] = np.std(mc_predictions[k,:])
    print(pred_mean)
    print(pred_std)
    plot_with_uncertainty(numOfUnits, pred_mean, pred_std)

# Plots the RUL predictions with uncertainty

def plot_with_uncertainty(numOfUnits, pred_mean, pred_std):
    actual_rul=np.array(pd.read_csv('PM_truth.txt',sep=' ',header=None).
        drop([1],axis=1))
    actual_rul = actual_rul[:numOfUnits]
    plt.plot(np.arange(numOfUnits), pred_mean, 'r-', label='Predictive mean')
    plt.plot(np.arange(numOfUnits), actual_rul, 'b', label='Training data')
    plt.fill_between(np.arange(numOfUnits),
                    pred_mean + 2 * pred_std,
                    pred_mean - 2 * pred_std,
                    alpha=0.5, label='Epistemic uncertainty')
    plt.title('Prediction')
    plt.legend()
    plt.show()


# Classify the state of component within next 30 days (either failed or funcitoning)

def binary_classification(num_epochs, load_ex_model):
    if load_ex_model == 0:
        model = Sequential()
        model.add(SimpleRNN(
                    input_shape=(timestamp, nb_features),
                    units=100,
                    return_sequences=True, activation='tanh'))
        model.add(Dropout(0.2, training = True))
        model.add(SimpleRNN(
                    units=50,
                    return_sequences=False, activation='tanh'))
        model.add(Dropout(0.2, training = True))
        model.add(Dense(units=1, activation='sigmoid'))
        model.compile(loss='binary_crossentropy', optimizer='adam',
            metrics=['accuracy'])
        model.fit(X_train, y_train, epochs=num_epochs, batch_size=200,
            validation_split=0.2, verbose=2,
            callbacks=[tensorboard, tensorboard2])
        #EarlyStopping(monitor='val_loss', min_delta=0, patience=0,
            verbose=0, mode='auto')
        model.save("classification_model_saved")
    else:
        model = load_model("classification_model_saved")

    model.summary()
    scores = model.evaluate(X_train, y_train, verbose=1, batch_size=200)
    print('Accuracy: {}'.format(scores[1]))
    y_pred = model.predict_classes(X_test)
    print('Accuracy of model on test data: ',accuracy_score(y_test,y_pred))
    print('Confusion Matrix: \n',confusion_matrix(y_test,y_pred))
    for Id in range(1, 10):
        print('Probability that machine {id} will fail within {period} days:
            '.format(id=Id, period=period), prob_failure(Id, model))


# RUL prediction using GRU recurrent network, random droupout

def RUL_regression(num_epochs, numOfUnits, load_ex_model):
    if load_ex_model == 0:
        model = Sequential()
        model.add(GRU(
                    input_shape=(timestamp, nb_features),
                    units=100,
                    return_sequences=True, activation='tanh'))
        model.add(Dropout(0.2, training = True))
        model.add(GRU(
                    units=50,
                    return_sequences=False, activation='tanh'))
        model.add(Dropout(0.2, training = True))
        model.add(Dense(units=1, activation='relu'))
        model.compile(loss='mean_squared_error', optimizer= RMSprop(lr = 0.01),
            metrics=['mse'])
        model.fit(X_train, y_train, epochs=num_epochs, batch_size=200,
            validation_split=0.2, verbose=2,
            callbacks=[tensorboard, tensorboard2])
        #EarlyStopping(monitor='val_loss', min_delta=0, patience=0,
            verbose=0, mode='auto')
        model.save("regression_model_saved")
    else:
        model = load_model("regression_model_saved")

    model.summary()
    scores = model.evaluate(X_train, y_train, verbose=1, batch_size=200)
    print('Accumulated loss: {}'.format(scores[1]))
    y_pred = model.predict(X_test)
    #print('Accuracy of model on test data: ',accuracy_score(y_test,y_pred))
    #print('Confusion Matrix: \n',confusion_matrix(y_test,y_pred))
    predict_RULs(numOfUnits, model)


RUL_regression(num_epochs=40, numOfUnits = 100, load_ex_model = 0)
```

Bibliography

Accorsi, Riccardo et al. (2017). “Data mining and machine learning for condition-based maintenance”. In: *Procedia Manufacturing* 11, pp. 1153–1161.

Adebiyi, Ayodele Ariyo, Aderemi Oluyinka Adewumi, and Charles Korede Ayo (2014). “Comparison of ARIMA and artificial neural networks models for stock price prediction”. In: *Journal of Applied Mathematics* 2014.

Babenko, Boris (2008). “Multiple instance learning: algorithms and applications”. In: *View Article PubMed/NCBI Google Scholar*, pp. 1–19.

Baraldi, Piero et al. (2013). “Model-based and data-driven prognostics under different available information”. In: *Probabilistic Engineering Mechanics* 32, pp. 66–79.

Barros, Anne (Sept. 2019). *Compendium, TPK4450 - Data Driven Prognostic and Predictive Maintenance.*

Basora, Luis, Xavier Olive, and Thomas Dubot (2019). “Recent Advances in Anomaly Detection Methods Applied to Aviation”. In: *Aerospace* 6.11, p. 117.

Box, George EP et al. (2015). *Time series analysis: forecasting and control.* John Wiley & Sons.

Brownlee, Jason (2016). “Supervised and unsupervised machine learning algorithms”. In: *Machine Learning Mastery* 16.03.

Cadenas, Erasmo and Wilfrido Rivera (2010). “Wind speed forecasting in three different regions of Mexico, using a hybrid ARIMA–ANN model”. In: *Renewable Energy* 35.12, pp. 2732–2738.

Cao, Li-Juan and Francis Eng Hock Tay (2003). “Support vector machine with adaptive parameters in financial time series forecasting”. In: *IEEE Transactions on neural networks* 14.6, pp. 1506–1518.

Chang, Fu-Kuo (2013). *Structural Health Monitoring 2013: A Roadmap to Intelligent Structures: Proceedings of the Ninth International Workshop on Structural Health Monitoring, September 10–12, 2013.* DEStech Publications, Inc.

Chen, Ling and Xu Lai (2011). “Comparison between ARIMA and ANN models used in short-term wind speed forecasting”. In: *2011 Asia-Pacific Power and Energy Engineering Conference.* IEEE, pp. 1–4.

Chollet, François (2018). *Deep Learning with Python.* MITP-Verlags GmbH & Co. KG.

Coble, Jamie B and J Wesley Hines (2008). “Prognostic algorithm categorization with PHM challenge application”. In: *2008 International Conference on Prognostics and Health Management*. IEEE, pp. 1–11.

El Naqa, Issam and Martin J Murphy (2015). “What is machine learning?” In: *Machine Learning in Radiation Oncology*. Springer, pp. 3–11.

Elasha, Faris et al. (2019). “Prognosis of a Wind Turbine Gearbox Bearing Using Supervised Machine Learning”. In: *Sensors* 19.14, p. 3092.

Ensari, Tolga et al. (2019). “Overview of Machine Learning Approaches for Wireless Communication”. In: *Next-Generation Wireless Networks Meet Advanced Machine Learning Applications*. IGI Global, pp. 123–140.

Erfani, Sarah M et al. (2016). “High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning”. In: *Pattern Recognition* 58, pp. 121–134.

Fahimifard, SM et al. (2009). “Comparison of ANFIS, ANN, GARCH and ARIMA techniques to exchange rate forecasting”. In: *Journal of Applied Sciences* 9.20, pp. 3641–3651.

Farrar, Charles R and Nick AJ Lieven (2006). “Damage prognosis: the future of structural health monitoring”. In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 365.1851, pp. 623–632.

Farrar, Charles R and Keith Worden (2012). *Structural Health Monitoring.: A Machine Learning Perspective*. John Wiley & Sons.

Fink, Olga, Enrico Zio, and Ulrich Weidmann (2015). “A classification framework for predicting components’ remaining useful life based on discrete-event diagnostic data”. In: *IEEE Transactions on Reliability* 64.3, pp. 1049–1056.

Fu, Rui, Zuo Zhang, and Li Li (2016). “Using LSTM and GRU neural network methods for traffic flow prediction”. In: *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*. IEEE, pp. 324–328.

Gal, Yarin and Zoubin Ghahramani (2016). “Dropout as a bayesian approximation: Representing model uncertainty in deep learning”. In: *international conference on machine learning*, pp. 1050–1059.

Gers, Felix A, Jürgen Schmidhuber, and Fred Cummins (1999). “Learning to forget: Continual prediction with LSTM”. In:

Gutschi, Clemens et al. (2019). “Log-based predictive maintenance in discrete parts manufacturing”. In: *Procedia CIRP* 79, pp. 528–533.

Hochreiter, Sepp (1998). “The vanishing gradient problem during learning recurrent neural nets and problem solutions”. In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6.02, pp. 107–116.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). “Long short-term memory”. In: *Neural computation* 9.8, pp. 1735–1780.

Hu, Chao et al. (2012). "Ensemble of data-driven prognostic algorithms for robust prediction of remaining useful life". In: *Reliability Engineering & System Safety* 103, pp. 120–135.

Huang, Guang-Bin, Xiaojian Ding, and Hongming Zhou (2010). "Optimization method based extreme learning machine for classification". In: *Neurocomputing* 74.1-3, pp. 155–163.

Jain, Dinesh and PK Srijith (2019). "Uncertainity quantification in Convolutional Deep Gaussian Process". PhD thesis. Indian institute of technology Hyderabad.

Jardine, Andrew KS, Daming Lin, and Dragan Banjevic (2006). "A review on machinery diagnostics and prognostics implementing condition-based maintenance". In: *Mechanical systems and signal processing* 20.7, pp. 1483–1510.

Juszczak, Piotr, D Tax, and Robert PW Duin (2002). "Feature scaling in support vector data description". In: *Proc. ASCI*. Citeseer, pp. 95–102.

Karbasi, Ali Reza, Somayeh Shirzadi Laskukalayeh, and Seiad Mohammad Fahimifard (2009). *Comparison of NNARX, ANN and ARIMA Techniques to Poultry Retail Price Forecasting*. Tech. rep.

Kendall, Alex and Yarin Gal (2017). "What uncertainties do we need in bayesian deep learning for computer vision?" In: *Advances in neural information processing systems*, pp. 5574–5584.

Kim, Kyoung-jae (2003). "Financial time series forecasting using support vector machines". In: *Neurocomputing* 55.1-2, pp. 307–319.

Ko, JM and YQ Ni (2005). "Technology developments in structural health monitoring of large-scale bridges". In: *Engineering structures* 27.12, pp. 1715–1725.

Korvesis, Panagiotis (2017). "Machine Learning for Predictive Maintenance in Aviation". PhD thesis.

Korvesis, Panagiotis, Stephane Besseau, and Michalis Vazirgiannis (2018). "Predictive Maintenance in Aviation: Failure Prediction from Post-Flight Reports". In: *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, pp. 1414–1422.

Krishna, K and Narasimha M Murty (1999). "Genetic K-means algorithm". In: *IEEE Transactions on Systems Man And Cybernetics-Part B: Cybernetics* 29.3, pp. 433–439.

Li, Kun-Lun et al. (2003). "Improving one-class SVM for anomaly detection". In: *Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No. 03EX693)*. Vol. 5. IEEE, pp. 3077–3081.

Liang, P and NK Bose (1996). "Neural network fundamentals with graphs, algorithms and applications". In: *Mac Graw-Hill*.

Liu, Hui, Hong-qi Tian, and Yan-fei Li (2012). "Comparison of two new ARIMA-ANN and ARIMA-Kalman hybrid methods for wind speed prediction". In: *Applied Energy* 98, pp. 415–424.

Liu, Qiong and Ying Wu (2012). "Supervised Learning". In: *Encyclopedia of the Sciences of Learning*. Ed. by Norbert M. Seel. Boston, MA: Springer US, pp. 3243–3245. ISBN: 978-1-4419-1428-

6. DOI: `10.1007/978-1-4419-1428-6_451`. URL: `https://doi.org/10.1007/978-1-4419-1428-6_451`.

Lorberfeld, Audrey (2019). *Machine Learning Algorithms In Layman's Terms, Part 1*. URL: `https://towardsdatascience.com/machine-learning-algorithms-in-laymans-terms-part-1-d0368d769a7b` (visited on 11/08/2019).

Mahamad, Abd Kadir, Sharifah Saon, and Takashi Hiyama (2010). "Predicting remaining useful life of rotating machinery based artificial neural network". In: *Computers & Mathematics with Applications* 60.4, pp. 1078–1087.

Martin, K.F (1994). "A review by discussion of condition monitoring and fault-diagnosis in machine-tools". In: *International Journal of Machine Tools and Manufacture* 34, pp. 527–551.

Mitchell, Thomas M. (1997). *Machine Learning*. 1st ed. New York, NY, USA: McGraw-Hill, Inc. ISBN: 0070428077, 9780070428072.

Monostori, László (2003). "AI and machine learning techniques for managing complexity, changes and uncertainties in manufacturing". In: *Engineering applications of artificial intelligence* 16.4, pp. 277–291.

Moré, Jorge J (1978). "The Levenberg-Marquardt algorithm: implementation and theory". In: *Numerical analysis*. Springer, pp. 105–116.

Mukherjee, Indrajit and Srikanta Routroy (2012). "Comparing the performance of neural networks developed by using Levenberg–Marquardt and Quasi-Newton with the gradient descent algorithm for modelling a multiple response grinding process". In: *Expert Systems with Applications* 39.3, pp. 2397–2407.

Nguyen, Khanh TP and Kamal Medjaher (2019). "A new dynamic predictive maintenance framework using deep learning for failure prognostics". In: *Reliability Engineering & System Safety* 188, pp. 251–262.

Peng, Zhang and Zhou Liang (2018). "Anomaly Detection with Changing Cluster Centers". In: *International Conference on Advanced Data Mining and Applications*. Springer, pp. 42–54.

Pham, DT and AA Afify (2005). "Machine-learning techniques and their applications in manufacturing". In: *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* 219.5, pp. 395–412.

Rabatel, Julien, Sandra Bringay, and Pascal Poncelet (2011). "Anomaly detection in monitoring sensor data for preventive maintenance". In: *Expert Systems with Applications* 38.6, pp. 7003–7015.

Ramakrishnan, AG et al. (2018). "Using Monte Carlo dropout for non-stationary noise reduction from speech". In: *arXiv preprint arXiv:1808.09432*.

Sankararaman, Shankar and Kai Goebel (2013). "Why is the remaining useful life prediction uncertain". In: *Annual conference of the prognostics and health management society*. Vol. 2013.

Saranga, Haritha and Jezdimir Knezevic (2001). “Reliability prediction for condition-based maintained systems”. In: *Reliability Engineering & System Safety* 71.2, pp. 219–224.

Schmidhuber, Jürgen, F Gers, and Douglas Eck (2002). “Learning nonregular languages: A comparison of simple recurrent networks and LSTM”. In: *Neural computation* 14.9, pp. 2039–2041.

Senapaty, Goutam and U Sathish Rao (2018). “Vibration based condition monitoring of rotating machinery”. In: *MATEC Web of Conferences*. Vol. 144. EDP Sciences, p. 01021.

Shi, Zhongzhi (2011). “Support Vector Machine”. eng. In: *Advanced Artificial Intelligence*. World Scientific Publishing Co. Pte. Ltd., pp. 309–327. ISBN: 9789814291354.

Sipos, Ruben et al. (2014). “Log-based predictive maintenance”. In: *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, pp. 1867–1876.

Srivastava, Nitish et al. (2014). “Dropout: a simple way to prevent neural networks from overfitting”. In: *The journal of machine learning research* 15.1, pp. 1929–1958.

Steyerberg, Ewout W et al. (2013). “Prognosis Research Strategy (PROGRESS) 3: prognostic model research”. In: *PLoS medicine* 10.2, e1001381.

Susto, Gian Antonio et al. (2014). “Machine learning for predictive maintenance: A multiple classifier approach”. In: *IEEE Transactions on Industrial Informatics* 11.3, pp. 812–820.

Tay, Francis EH and Lijuan Cao (2001). “Application of support vector machines in financial time series forecasting”. In: *omega* 29.4, pp. 309–317.

Tobon-Mejia, Diego Alejandro et al. (2012). “A data-driven failure prognostics method based on mixture of Gaussians hidden Markov models”. In: *IEEE Transactions on reliability* 61.2, pp. 491–503.

Vachtsevanos, George J et al. (2006). *Intelligent fault diagnosis and prognosis for engineering systems*. Vol. 456. Wiley Hoboken.

Voyant, Cyril et al. (2017). “Machine learning methods for solar radiation forecasting: A review”. In: *Renewable Energy* 105, pp. 569–582.

Wang, Tianyi et al. (2008). “A similarity-based prognostics approach for remaining useful life estimation of engineered systems”. In: *2008 international conference on prognostics and health management*. IEEE, pp. 1–6.

Wold, Svante, Kim Esbensen, and Paul Geladi (1987). “Principal component analysis”. In: *Chemometrics and intelligent laboratory systems* 2.1-3, pp. 37–52.

Wuest, Thorsten et al. (2016). “Machine learning in manufacturing: advantages, challenges, and applications”. In: *Production & Manufacturing Research* 4.1, pp. 23–45. DOI: `10.1080/21693277.2016.1192517`. eprint: `https://doi.org/10.1080/21693277.2016.1192517`. URL: `https://doi.org/10.1080/21693277.2016.1192517`.

Yang, o-Suk and Achmad Widodo (2010). *Introduction of intelligent machine fault diagnosis and prognosis*. Nova Science Publishers, Incorporated.

Yildirim, Hasan and M Revan Özkale (2019). "The performance of ELM based ridge regression via the regularization parameters". In: *Expert Systems with Applications* 134, pp. 225–233.

Zhang, Cha and Yunqian Ma (2012). *Ensemble machine learning: methods and applications*. Springer.

Zhang, G Peter (2003). "Time series forecasting using a hybrid ARIMA and neural network model". In: *Neurocomputing* 50, pp. 159–175.

Zhang, Guoqiang, B Eddy Patuwo, and Michael Y Hu (1998). "Forecasting with artificial neural networks:: The state of the art". In: *International journal of forecasting* 14.1, pp. 35–62.

Zhang, Ke et al. (2016). "Automated IT system failure prediction: A deep learning approach". In: *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, pp. 1291–1300.

Zhang, Min-Ling and Zhi-Hua Zhou (2004). "Improve multi-instance neural networks through feature selection". In: *Neural Processing Letters* 19.1, pp. 1–10.