FACULTY OF INFORMATION TECHNOLOGY CTU IN PRAGUE

ASSIGNMENT OF MASTER’S THESIS

Title:	Application of Artificial Intelligence Techniques in Predictive Maintenance
Student:	Bc. Jan Lukány
Supervisor:	Ing. Tomáš Borovička
Study Programme:	Informatics
Study Branch:	Knowledge Engineering
Department:	Department of Applied Mathematics
Validity:	Until the end of summer semester 2020/21

Instructions

There exist multiple approaches to predictive maintenance (PdM) problems each having specific data requirements and use cases. Nowadays, these problems can be solved using artificial intelligence (AI) techniques. The goals of this thesis are to:
- Review common approaches to PdM, including fault detection, fault prediction, remaining useful life prediction and anomaly detection, and their evaluation metrics.
- Review several most used AI algorithms for each of the PdM approaches from both deep learning and classical machine learning.
- Experimentally compare the evaluation metrics on several publicly available datasets using the reviewed algorithms. Focus on the practical application.

References

Will be provided by the supervisor.

Ing. Karel Klouda, Ph.D.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague January 22, 2020

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master’s thesis

Application of Artificial Intelligence in Predictive Maintenance

Bc. Jan Lukány

Department of Applied Mathematics
Supervisor: Ing. Tomáš Borovička

May 28, 2020

Acknowledgements

I would like to express sincere gratitude to my supervisor Ing. Tomáš Borovička for his guidance and mentorship over several past years. I would also like to thank all my colleagues at Datamole. Finally, I would like to thank my family, my parents and especially my future wife, Barbora, for their endless support during my studies.

Declaration

In Prague on May 28, 2020 …………………

Citation of this thesis

Lukány, Jan. *Application of Artificial Intelligence in Predictive Maintenance.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2020.

Abstrakt

Prediktivní údržba je strategie plánování údržby, při níž je údržba naplánována pokud subjekt jeví známky závady nebo je pravděpodobné, že brzy dojde k poruše. Prediktivní údržba snižuje náklady a zabraňuje prostojům ve srovnání s klasickými strategiemi preventivní a reaktivní údržby. Prediktivní údržba může být realizována použitím technik umělé inteligence k vytvoření modelu, který zdravotní stav subjektu na základě dat získaných monitorováním jeho stavu. Existují však různé přístupy k prediktivní údržbě jako detekce závady, predikce poruch a predikce zbývající užitné životnosti, z nichž každý má odlišné požadavky na data a má jiné cíle. Každý z těchto přístupů využívá jiné techniky umělé inteligence a kvalita modelů vytvořených dle těchto přístupů by měla být hodnocena dle jiných metrik. Tato diplomová práce poskytuje přehled přístupů k prediktivní údržbě a pomáhá tak odborníkům zvolit vhodný přístup, techniku umělé inteligence a správnou hodnoticí metriku pro jejich problém.

Klíčová slova prediktivní údržba, umělá inteligence, detekce závad, predikce poruch, predikce zbývající užitné životnosti, monitorování stavu

Abstract

Predictive maintenance (PdM) is a maintenance strategy where the maintenance actions are scheduled only when the subject is malfunctioning or is likely to fail soon. PdM reduces costs and prevents downtime in comparison to classical preventive and reactive maintenance strategies. PdM can be realized by using artificial intelligence (AI) techniques to build a model that predicts health state of the subject based on its condition monitoring data. However, there exist various approaches to PdM including fault detection, failure prediction and remaining useful life prediction, each having different data requirements and goals. Each of the approaches utilizes different AI techniques and should be evaluated using different evaluation metrics. This thesis provides an overview of the approaches to PdM to help the practitioners choose a suitable approach, AI technique and evaluation metric for their problem at hand.

Keywords predictive maintenance, artificial intelligence, fault detection, anomaly detection, failure prediction, remaining useful life prediction, condition monitoring

Contents

List of Figures

List of Tables

Introduction

Motivation

Predictive maintenance (PdM) is a maintenance strategy where the goal is to monitor and analyze condition of a subject in order to plan maintenance actions at times when the subject suffers from a fault or when there is an increased probability that the subject will fail in near future. Such maintenance strategy can significantly reduce costs and possible downtime caused by failures in comparison with other strategies such as corrective or preventive where the maintenance actions are scheduled only when the machinery fails, and thus needs a correction, or are scheduled at regular intervals.

The condition monitoring is done by collecting various kinds of data that can contain information about the health state of the subject. The analysis can be then done by building a predictive model that is, given condition monitoring data, capable of predicting whether the subject is faulty or estimating when a failure will occur. Nowadays, such PdM models can be built utilizing artificial intelligence (AI), more specifically machine learning (ML), techniques where the models are trained on condition monitoring and health data of multiple subjects. Depending on what type of condition monitoring data is available, various ML modeling techniques can be used.

A crucial part of PdM is a performance evaluation of the built model, i.e. estimation how the model will perform in real-world. The performance evaluation has two major goals. The first goal is that it should serve as a way how to choose the best performing model when building models with different parameters or ML algorithms. The second goal is that the performance evaluation should be intuitively interpretable — e.g. how much in advance is the model able to predict a failure or how often the model predicts false alarms. As there exist various evaluation metrics which can be used for every modeling approach a good overview of different evaluation metrics and their advantages and disadvantages is crucial for a success of PdM project in industry.

Related Work

Predictive maintenance has drawn huge attention in both scientific and industrial research over the past two decades. Numerous scientific articles describing novel AI approaches to PdM as well as many articles describing the application of PdM in various domain such as predicting failures in wind turbines, hard drives, high-speed trains or power plants has been published in past years [17–23]. There have been published multiple reviews and surveys on predictive maintenance systems, purposes and different approaches [3, 5, 24–26]. Some works specifically focus on the application of various approaches of artificial intelligence and machine learning in predictive maintenance [27–29] while other works propose novel or adjusted evaluation metrics for the individual approaches [12, 30–32]. However, to our knowledge, there is no work that would provide an overview of multiple ML-based modeling approaches and would focus at the same time on comparison of the different evaluation metrics.

Goals

The goals of this thesis are to:

- give an introduction to the problematics of PdM;
- provide an overview of several different ML-based modeling approaches used for building PdM models;
- describe different evaluation metrics that can be used to assess the performance of the models built by different modeling approaches;
- compare and discuss the practical application of the different evaluation metrics by conducting experiments on real-world data sets.

Organization of the Thesis

This thesis is organized as follows. In Chapter 1 we provide a minimal theoretical background of ML including the classical machine learning tasks and their evaluation metrics. In Chapter 2 we provide an introduction to PdM in context of different maintenance strategies and we describe typical condition monitoring data used for building a PdM model. In Chapter 3 we review different approaches to PdM utilizing ML techniques and we describe how the built PdM models can be evaluated. Finally, in Chapter 4 we conduct experiments where we demonstrate the modeling approaches on real-world data sets, we compare their evaluation metrics and we discuss the metrics' practical application.

Chapter 1

Machine Learning Background

Machine learning (ML) is a an area of AI that studies computer algorithms that improve through experience. In this chapter we provide a minimal theoretical background of machine learning necessary for the rest of this thesis. The content of this chapter can be, with a few exceptions, considered as a common knowledge. Therefore, we cite only where we deem it necessary or where we use direct definitions from literature. As we provide only the minimal theoretical background we refer to [33–35] for a comprehensive overview of machine learning and related fields.

In Section 1.1 we describe three different types of ML algorithms — supervised, unsupervised and semi-supervised. In Section 1.2, we describe three machine learning problems — classification, regression and anomaly detection. In Section 1.3 we describe several ML models. Finally, in Section 1.4, we describe how to evaluate and select a machine learning model.

1.1 Types of Machine Learning Algorithms

There exist four main types of machine learning algorithms: supervised learning, unsupervised learning, semi-supervised learning and reinforcement-learning. In this thesis, we use especially the first three of them and we describe them below.

Supervised learning Supervised learning algorithm learns from a set of labeled samples and builds models that can predict label for new unseen samples.

Unsupervised learning Unsupervised learning ML algorithms consists in learning interesting or meaningful structures from a set of unlabeled samples. They help to understand the data.

Semi-supervised learning Semi-supervised machine learning is a combination of supervised and unsupervised learning. It makes use of both labeled and unlabeled samples to learn the relationship between the features and the target variable. Having both labeled and unlabeled samples is a common problem in practice — e.g. we can have medical data about lots of patients but we might have only a small portion of them labeled (e.g. whether they were sick or not).

1.2 Machine Learning Problems

1.2.1 Classification

Classification is a ML problem of where the labels, the target variables, of the samples are categorical. A problem of diagnosing whether a patient suffers from a disease based on its health condition is an example of a classification problem. It is solved by supervised learning algorithms. Classification can be divided into a binary and a multiclass, i.e. predicting two classes or multiple classes, respectively. Many methods for classification are developed for binary classification. Therefore, multiclass classification can be regarded as its extension. In case of binary classification, the two classes are commonly named as positive and negative and the model's predictions can be thus either positive, i.e. belongs to a positive class, or negative, i.e. belongs to a negative class.

Though the target variable is a category, a class, the classification can be done as predicting a probability of a sample belonging to the category. For example the model can predict that a probability that a patient is sick is 0.8 (and thus the probability that he/she isn't sick is 0.2 %). The final prediction of the category then can be done by setting a decision threshold which defines the minimal probability necessary for the sample to be considered positive. A typical default threshold is 0.5.

1.2.2 Anomaly Detection

Anomaly detection is a machine learning problem where the goal is to identify the most anomalous samples. It is typically solved by unsupervised ml algorithms. The detection of anomalies is typically done by predicting some kind of anomaly score for each sample (e.g. distance from mean of the distribution of features in the training data) and setting a threshold that marks the samples with higher score than the threshold as anomalous.

1.2.3 Regression

Regression is a problem of identifying a relationship between the features and a continuous target variable. For example predicting price of houses based on their features like location, size or number of rooms is a regression problem.

1.3 Machine Learning Models

In this section, we describe three examples of ML models. We provide only brief description that is essential for the rest of this thesis.

1.3.1 Decision Tree

Decision tree is a supervised learning algorithm which can be used for both classification and regression problems. It consists in constructing a set of rules in a form of a tree where the leaves of the tree are assigned the values of a target variable (either class or a continuous variable). For example in case of patients diagnosis, the rules can be ”Has temperature higher than 37 degrees?” or ”Has difficult breathing?”. The classification of a sample is then done by traversing through the tree, following the rules, and assigning it the value of the leaf where the sample ends. The primary objective in constructing the decision tree is that the rules should describe the data as best as possible — that is done for example by finding such rules that minimize the entropy of the data when the data are divided by the rule.

There exists plenty of variants of decision tree and their extensions. Random forest is a decision tree based algorithm where multiple decision trees are built and the output, the target variable, is then a mode or a mean of the outputs of the trees. One of the currently best performing variant of decision trees is an algorithm called extreme gradient boosted trees [36]. It consists in building a large numbers of low complexity trees (weak learners) so that each tree predicts a length of a move in a direction of a gradient of a predefined loss function. Combining the predictions of these trees then leads to a single continuous predicted target variable (can be in a form of class probability).

1.3.2 SVM

Support-vector machine (SVM) is a supervised machine learning algorithm introduced by Vapnik [37] that is used for binary classification problem and can be extended to solve regression problem, in that case being called SVR. The main idea of SVM is to transform the samples into a higher dimensional space and find a hyperplane that best separates the two classes. The samples on the margins of the hyperplane are called support vectors, hence the name.

1.3.3 Artificial neural networks

()s a computing system inspired by human brain and can be used to solve classification, regression and anomaly detection problems. It consists of a set of connected artificial neurons, cells, that can transmit information through the connections. The transmitted information is in a form of a real number whose is given by a sum of the neurons inputs, i.e. the information transmitted to it by other neurons, and some non-linear function. The neurons are typically structured in layers. The input, the features, are then typically given as an input information to the neuron in a so called input layer while the output, the target variable(s), is then an output of the so called output layer of neurons. Each neuron can have a weight that increases or decreases the amount of information transferred. The training of a ANN then consists in adjusting weights of the individual neurons so that the outputs of ANN is closer to the desired output. For more details work on how the ANN are trained we refer to [34].

Each layer of ANN can perform different transformations and can have different number of neurons. The way how the neurons are organized into the layers and how they are connected to each other is called an ANN architecture. Below we provide an overview of common ANN architectures than we will refer to in the the rest of this thesis.

Feedforward One of the basic architectures of ANN is a feedforward ANN. A feedforward ANN is such ANN where the connections between the neurons do not form a cycle, i.e. the information is transferred only in forward direction from the input layer to the output layer.

Recurrent networks Recurrent neural networks are derived from feedforward networks where, however, the connections can be cyclic. Recurrent neural networks can learn not only on single points but also on a series of data such as time series, sequences of words or videos. One of the most used recurrent neural networks is an long short-term memory (LSTM) network.

Convolutional networks Convolutional neural network are feedforward neural network that consist of layers where the information passed to neurons in next layer is modified by convolution operation with a filter composed of weights. It is commonly applied in problems where the input is in a form of an image. The filters then can have weights that for example detect edges and, when multiple convolutional layers are employed, even complex patterns can be recognized.

Autoencoders Autoencoders are type of ANN which are trained to reproduce the input to the output while internally representing the input in some compressed form, a code. One of the use cases for autoencoders is anomaly detection where the anomalies fed to the autoencoder are supposed to have a higher reconstruction error (difference between input and output) than the normal samples. The reconstruction error can thus be taken as the anomaly score.

1.4 Evaluation

Evaluation of a machine learning model consists in estimating how the model will perform on a randomly selected data, independent from the training data. Therefore, the evaluation typically consist in splitting the data set on a training and testing sets, using the training set to train the model and calculate evaluation metrics on a testing data set.

The evaluation results have two major goals: to interpret the model's performance (e.g. what is a probability that a sick patient will be detected) and to select the best performing model out of multiple different trained models. The evaluation metrics used for the performance interpretation and model selection can be different as for example some metrics might be difficult to interpret in the domain.

In this section we describe the various evaluation metrics used for both classification and regression problems[1] and then we briefly describe the process of model selection.

1.4.1 Evaluation Metrics for Classification

Predictions of a binary classification can be expressed by a confusion matrix:

		Actual	
		neg	pos
Predicted	neg	TN	FN
	pos	FP	TP
		N	P

where TP, FP, TN and FN stand for true positive, false positive, true negative and false negative, respectively. We also denote P and N as the number of total actual positives and negatives, respectively.

Four commonly used metrics for evaluation of classification performance are:

- *accuracy* $= \frac{\text{TP}+\text{TN}}{\text{P}+\text{N}}$ — a probability of a prediction being correct;
- *precision* $= \frac{\text{TP}}{\text{TP}+\text{FP}}$ — a probability that the actual label is positive when predicted as positive, e.g. a probability that a patient is actually sick when the model predicts he/she is sick;
- *recall* $= \frac{\text{TP}}{\text{P}}$ — a probability that an actual positive label is predicted as positive, also called a true positive rate (TPR), e.g. a probability that a patient is predicted as sick given that he/she actually is sick;

[1]note, that anomaly detection can be evaluated using classification metrics if we have a labeled testing data set

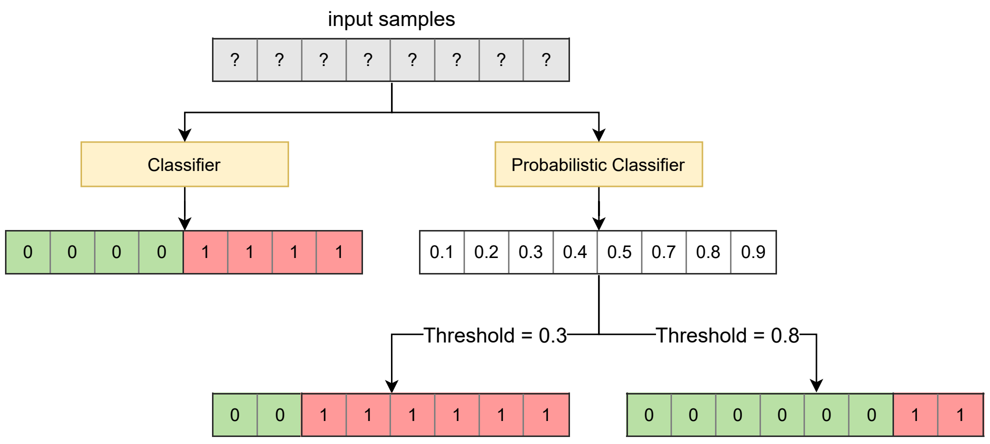


Figure 1.1: Predicting probabilities instead of classes

- *false positive rate (FPR)* $= \frac{\text{FP}}{\text{N}}$ — a probability of a negative sample being predicted as positive, e.g. a probability that a healthy patient is diagnosed as sick.

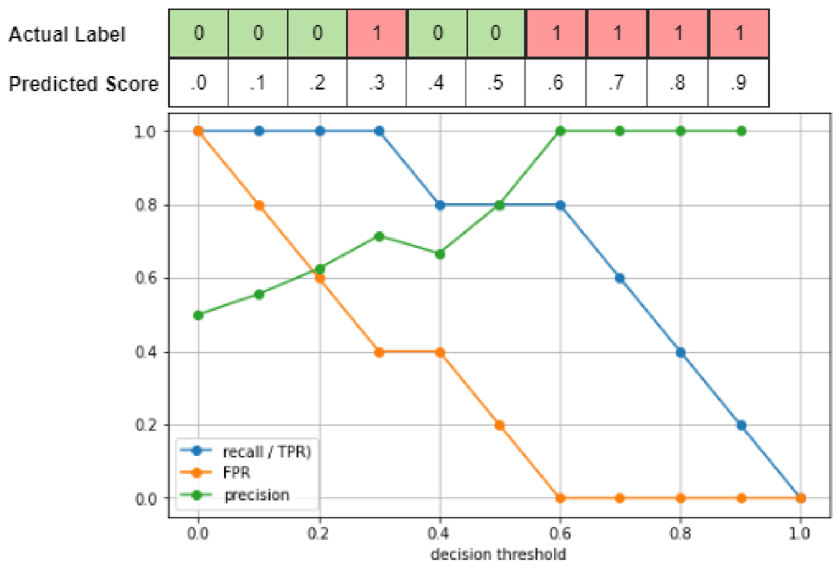
A model having a high recall might have a low precision (e.g. a model that predicts only positive predictions) and vice versa. Therefore, precision and recall are commonly expressed by calculating their harmonic mean. Such constructed metric is called an F1 score and is formally defined as

$$\text{F1 score} = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}.$$

Most of machine learning (ML) binary classification and anomaly detection algorithms are capable of predicting a score — a continuous variable like probability of belonging to the positive class or e.g. some measure of distance from the normal points in case of anomaly detectors. A classifier that predicts probabilities is commonly called probabilistic classifier. The actual classification (anomaly detection) is then done by setting a decision threshold — if the score is equal or greater than the decision threshold the prediction is positive and vice versa (as illustrated in Figure 1.1).

A common decision threshold for supervised (binary) classification algorithms that predict probability is 0.5 [38] which is typically where the F1 score is the highest. In anomaly detection, on the other hand, there is no universal threshold that can be set as the scores do not have the intuitive probabilistic interpretations. Moreover, it might happen, that FPs and FNs have each different severity. For example in a medical screening test it is wanted to have as few FNs[2] as possible even though that might yield many FPs. In other words, the medical screening tests should have a high recall and low precision

[2]sick patients diagnosed as healthy



Actual Label	0	0	0	1	0	0	1	1	1	1
Predicted Score	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9

Figure 1.2: Precision, recall and FPR over various decision thresholds

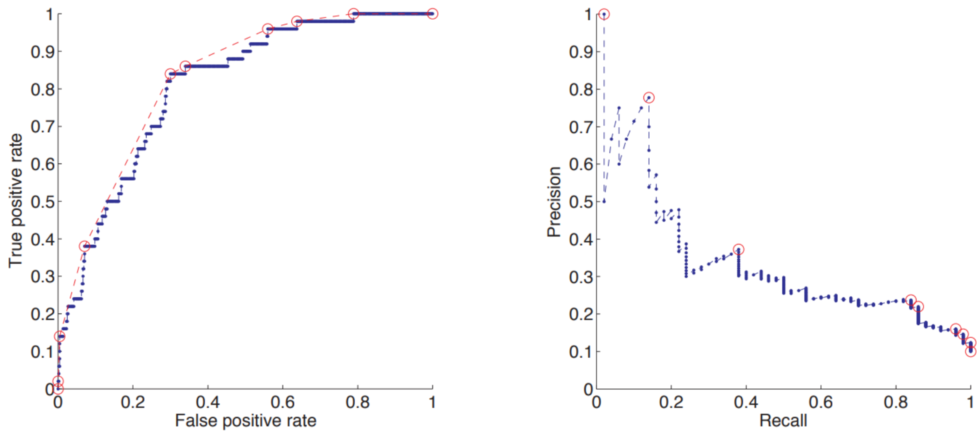


Figure 1.3: ROC curve (left) and corresponding PR curve (right) [1]

is tolerated. On the other hand, for example anti-virus systems shouldn't raise too many false alarms, i.e. when they identify something as a positive they should be certain about it. In other words, anti-virus systems should have a high precision while lower recall might be tolerated. Setting a higher decision threshold typically leads to higher precision (though not necessarily) whereas setting a lower decision leads to higher recall. Therefore, selecting a decision threshold should be made with a good domain knowledge.

One possible way how to analyze the models performance on various decision thresholds is visualizing precision, recall and FPR metrics over various decision thresholds as illustrated in Figure 1.2. However, such visualization is dependent on the actual range of decision thresholds which does not have to be in range $[0, 1]$. Therefore, receiver operating characteristic (ROC) and precision-recall (PR) curves are commonly used to visualize the performance over various thresholds. ROC curve is a plot of TPR (recall) over FPR as illustrated in the left part of Figure 1.3. PR curve is then a plot of precision against recall as illustrated in the right part of Figure 1.3.

ROC curve is non-decreasing — when increasing the threshold, both TPR and FPR either stay the same or increase. Moreover, ROC has an important property that it is possible to construct a model at any point on a line connecting two points on an ROC curve. This can be achieved by combining the predictions from the models corresponding to the two points, e.g. selecting half of the predictions from a model A and half of the predictions from a model B results in a model that has performance corresponding exactly to the point in the middle of the line connecting the two models points on an ROC curve. This results in an existence of a universal baseline in an ROC curve — a line connecting left lower and right upper corners which correspond to an always-negative model and an always-positive model.

PR curve, on the other hand, does not have any universal baseline. Instead, the baseline is different for every data set and corresponds to a horizontal line at precision equal to prevalence (π) — the ratio of positive samples in the data set. This baseline then corresponds to a performance of a random classifier. Moreover, the PR curve does not have the property of linear interpolation as ROC curve does. This is mainly caused by the fact that PR curve is neither (non-)decreasing nor (non-)increasing. That is because increasing the decision threshold might decrease precision (as seen in Figure 1.2 where the precision decreased with increasing threshold from 0.3 to 0.4). However, PR curve does have one big advantage over ROC — it is suitable for evaluating imbalanced data sets (data sets with low prevalence) as neither precision nor recall depend on the amount of true negatives.

A domain knowledge is required to select the right decision threshold. If the domain knowledge is not available though, it might be desirable to select a model that performs best at regardless of the chosen decision threshold and leave the decision threshold selection for later. For that an area under curve ROC (AUROC) $\in [0, 1]$ is typically used. AUROC has even a natural explanation — it estimates the probability that a randomly chosen positive is ranked higher by the model than a randomly chosen negative [39].

Maybe inspired by the AUROC, some researchers started using area under PR curve (AUPR) to evaluate models on imbalanced data sets. However, calculation of AUPR done via trapezodial rule (a common way how area under curve is calculated) is wrong as the points on the PR curve should not be linearly interpolated and selecting a model by AUPR might thus result in selecting a worse performing model [1]. To mitigate this problem, Flach et al. introduced precision-recall-gain (PRG) curve [1]. The main idea of PRG curves is to express the precision and recall in terms of gain over a baseline

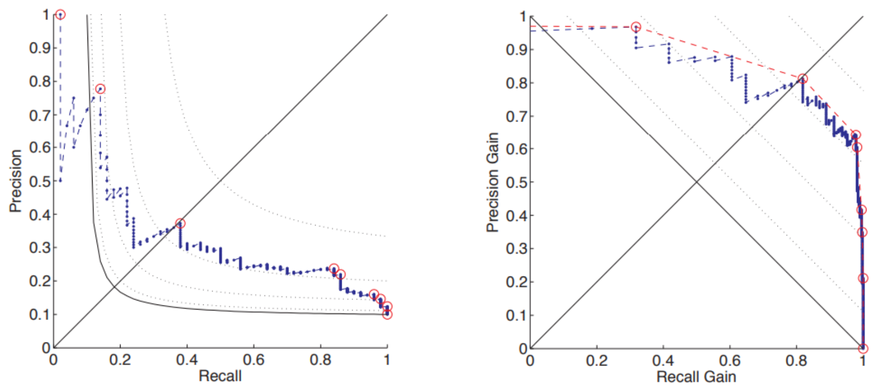


Figure 1.4: PR curve and a corresponding PRG curve [1]. The dotted lines represent F1 and F1-gain isometrics, respectively.

model — a model that predicts always positive predictions. By using harmonic scaling:

$$\frac{1/x - 1/\min}{1/\max - 1/\min} = \frac{\max(x - \min)}{(\max - \min)x}$$

and taking $\min = \pi$ and $\max = 1$ precision-gain and recall-gain are defined as [1]:

$$\text{precision-gain} = \frac{\text{precision} - \pi}{(1 - \pi)\text{precision}} = 1 - \frac{\pi}{1 - \pi}\frac{\text{FP}}{\text{TP}},$$

$$\text{recall-gain} = \frac{\text{recall} - \pi}{(1 - \pi)\text{recall}} = 1 - \frac{\pi}{1 - \pi}\frac{\text{FN}}{\text{TP}}.$$

A PRG curve is then a plot of precision-gain over recall-gain. Figure 1.4 illustrates a PR curve and a corresponding PRG curve. Calculating area under PRG curve (AUPRG) is then possible with a linear interpolation and is related to an expected F1 score [1].

1.4.2 Evaluation Metrics for Regression

Prediction made by a regression model is a continuous variable. Let us denote N the number of samples we are evaluating and y_i and $\hat{y}_i$ the actual and predicted value of the i-th sample. A standard metrics for evaluating regression include

$$\begin{aligned}\text{mean absolute error (MAE)} &= \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i),\\ \text{root mean squared error (RMSE)} &= \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_t - \hat{y}_i)^2},\\ \text{mean absolute percentage error (MAPE)} &= \frac{1}{N}\sum_{i=1}^{N}\left|\frac{y_t - \hat{y}_i}{y_i}\right|.\end{aligned}$$

MAE is metric that gives the same weight to all errors. RMSE gives more weight to high errors. MAPE, on the other hand, gives more weight to errors at low values as e.g. an error with $y = 100$ and $y = 150$ is equivalent to error $y = 1$ and $y = 1.5$ — the error is 0.5 (or 50 %).

1.4.3 Model Selection

Model selection is a process of selecting between either different machine learning algorithms (e.g. whether to use a decision tree or SVM) or selection of the best hyperparameters for a given model. The hyperparameters can be for example a maximal depth of a decision tree or a number of layers in an ANN.

The model selection then typically consists in training multiple models, evaluating them and choosing the one that performs the best. In order to avoid selecting a model that is overfitted to a certain kind of data cross-validation is often used.

1.4.4 Cross-validation

Cross-validation (CV) is a technique used to evaluate how a model will perform on an independent data set. In its basic form, CV consists in splitting a data set into multiple sets of same size called folds and performing multiple training and testing phases. In each phase one fold is selected as testing and the rest as training. The model is then build using the training folds and evaluated using the testing fold. A CV using K folds is commonly called a K-fold CV The output of the CV are then K scores where K is the amount of folds and each score corresponds to a testing score of one fold. A mean of the scores over the testing folds is then commonly calculated and it can serve as a primary metric for model selection.

Chapter 2

Introduction to Predictive Maintenance

In this chapter we provide an introduction to the problematic of PdM. In Section 2.1 we explain the motivation in context of other maintenance strategies. In Section 2.2, we describe condition monitoring, a fundamental process of PdM, which consists in gathering data that can help reveal the condition of the subject. Finally, in Section 2.3 we provide an introduction to different approaches to PdM, i.e. how can be condition monitoring data used to predict the condition of the subject.

2.1 Motivation

A life-cycle of industrial machinery consists of several stages with the operation stage usually being the longest stage of all [2], as illustrated in Figure 2.1. During this stage the machinery might develop a fault or may naturally degrade. Both a fault and a degradation have a negative effect on its health, i.e. its ability to operate. Moreover, the fault or the degradation can grow in severity over time and may lead to a failure [3, 4]. Figure 2.2 illustrates a difference between a fault and a failure. A failure may be either an inability of the subject to operate at all which causes downtime or it might be considered as reaching some threshold of permissible degradation, commonly called a failure threshold [3]. In both the cases the failure is a highly unwanted

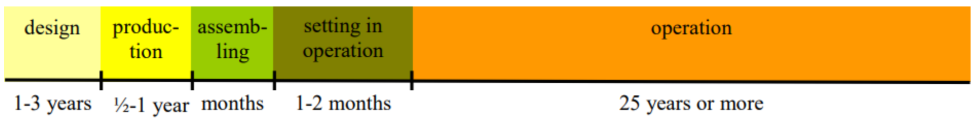
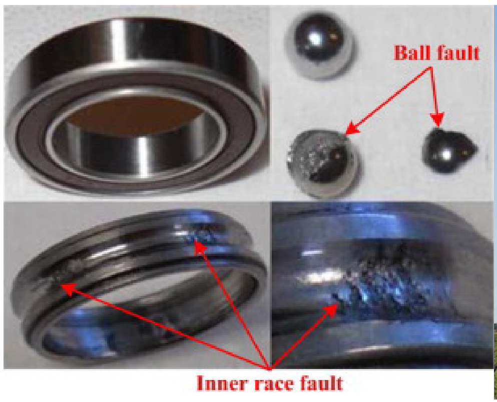
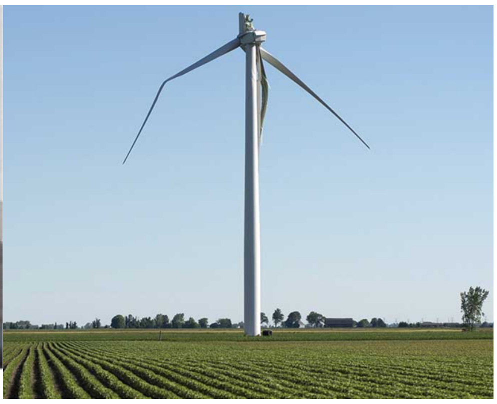


Figure 2.1: Machinery life stages [2].



(a) fault



(b) failure

Figure 2.2: The difference between a fault and a failure: (**a**) a fault of a bearing [3]; (**b**) a failure of a wind turbine [4].

event which decreases reliability and may costs a high amount of resources, both human and financial, to fix [40].

The industrial machinery is a typical example where faults, degradation and failures occur. However, it is definitely not limited to the industrial machinery. Hard drive can fail [20], network faults can occur [41] and, with a bit of exaggeration, even humans can suffer from a fault, can degrade and eventually fail, e.g. a hearth failures [42, 43]. Therefore, to express this generality we will stick to the term subject.

To preserve the health of the subject maintenance actions are performed during which an action that mitigates the fault or the degradation is executed, e.g. a replacement of a faulty part such as a bearing [40]. There exist two classical maintenance strategies: reactive (also called corrective) and preventive [5].

Reactive Maintenance Reactive maintenance strategy is performed as a reaction to a failure. It is sometimes also referred to as a corrective maintenance as a failed component/part is typically repaired or corrected [5]. Reactive maintenance strategy reduces the amount of maintenance actions (they are done only when absolutely needed). However, reactive maintenance requires high availability of the personnel responsible for the maintenance actions as the failure might happen e.g. in the middle of the night and, most importantly, and, most importantly, it does not prevent a failure — so there is either a downtime or unsafe operation.

Preventive Maintenance The second strategy called preventive maintenance is based on scheduling the maintenance actions at predefined intervals such as twice a year [5]. Preventive maintenance can significantly reduce the

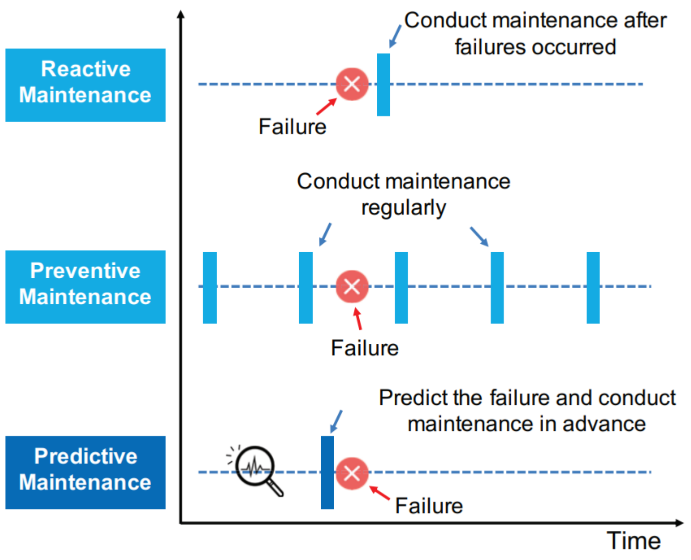

Reactive Maintenance
Conduct maintenance after failures occurred
Failure
Preventive Maintenance
Conduct maintenance regularly
Failure
Predictive Maintenance
Predict the failure and conduct maintenance in advance
Failure
Time

Figure 2.3: Maintenance plans of RM, PM and PdM [5].

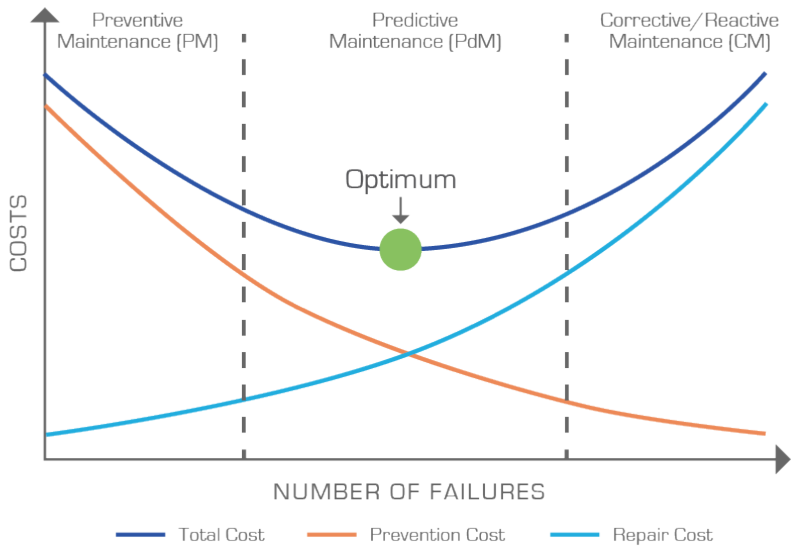


Figure 2.4: Costs of maintenance strategies [6].

risk of failures as the subject is regularly checked, but may schedule maintenance actions even when it is not necessary which increases maintenance costs [5].

Predictive Maintenance Predictive maintenance strategy aims for a compromise between the two classical strategies mentioned above by scheduling the maintenance only when the subject exhibits signs of a fault or degradation [5]. Figure 2.3 illustrates planning of maintenance actions according to a reactive, preventive and predictive strategies. Figure 2.4 illustrates the reduction of costs predictive maintenance brings. The main goal of predictive maintenance is to monitor and analyze the condition of the subject and provide the personnel responsible for the maintenance scheduling the information about the subject's condition so that a maintenance action can be scheduled when needed [5, 40].

2.2 Condition Monitoring

Condition monitoring is a process of collecting information that might reveal the health state of a subject [44]. The health state of the subject can be observed directly, e.g. the amount of wear of a cutting machine [10], or indirectly such as measuring operating conditions of a subject (e.g. outside temperature) or calculating time from last maintenance action [45]. In this section we describe various sources of the condition monitoring data.

2.2.1 Operational Settings

Operational settings are any subject specific settings such as load, speed, cycle number or current firmware version and may change over time [46]. Operational settings might affect how fast the subject degrades, e.g. a subject operating under higher loads than the rest of the subjects is likely to degrade sooner.

2.2.2 Environment data

Environment data include any information about the environment where the subject operates, the external factors. Common environment data include outside temperature, geolocation or season of the year. Environment data might be important predictors of health as they might have effect on how the subject operates. For example a compressor might have a higher energy consumption during winter than during summer, during which such high energy consumption might be considered as anomalous and thus might signify a fault. The outside temperature of the environment can also have significant effect on how fast the capacity of a lithium-ion battery degrade [47].

2.2.3 Sensor Data

Sensor data such as pressure, vibration or acoustic noise are one of the most commonly measured condition monitoring data [5]. The sensors can measure directly the subject, e.g. vibrations of rotating machinery such as pressure in a compressor, or they might measure the environment where the subject is

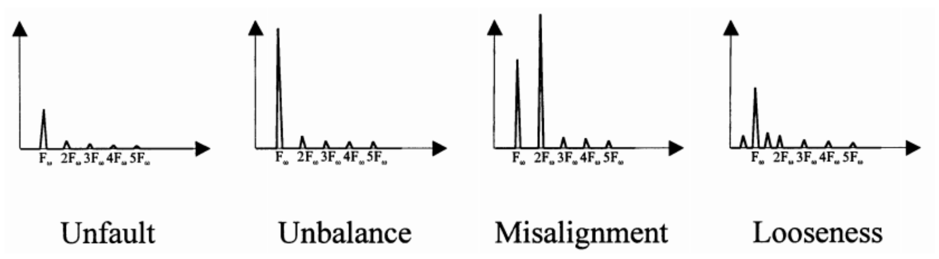


Figure 2.5: Four frequency spectra of rotating machinery vibration signals each representing different health state. On the x-axes are frequencies while on the y-axes are amplitudes. F is a driving frequency — the frequency equivalent to the speed of rotating.

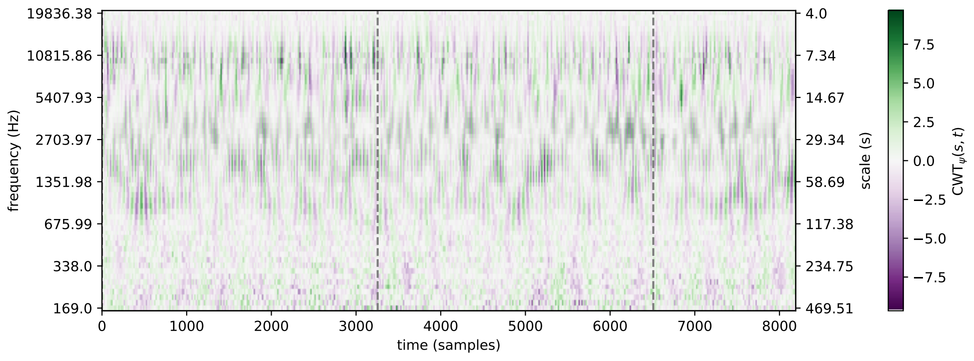
operating, thus being basically source of environment data mentioned above [2].

The sensor data are typically in a form of time series, sampled at periodic intervals. For sensor where the observed variable doesn't change as frequently (e.g. temperature) the sampling frequency can be relatively low such as one sample per hour. However, in case of e.g. vibration or acoustic signals measured on fast rotating machinery the sampling frequency is commonly in KHz, i.e. thousands of samples per second [48].

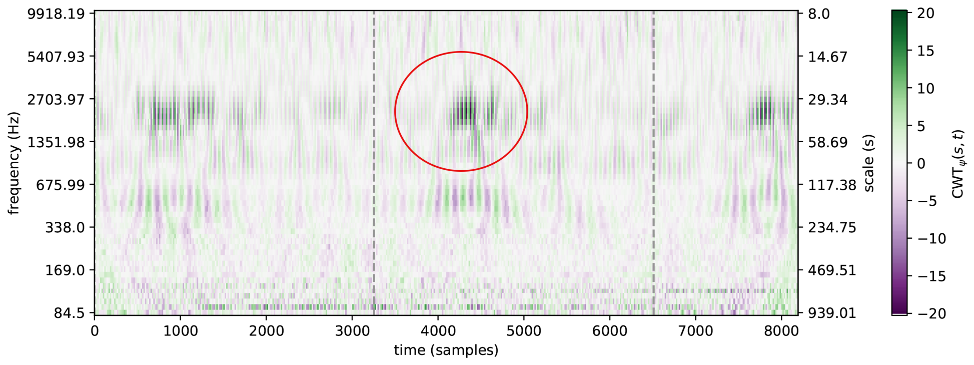
Sensor data can be used in their natural time representation, i.e. a waveform. However, there exist several preprocessing techniques that can transform the data into different representation where faults can be more easily revealed such as Fourier or Wavelet transforms which transform the data to frequency or time-frequency representations, respectively. These transformations are especially suitable for revealing faults in rotating machinery where an increase of amplitude at certain frequencies can signify a fault [7]. Figure 2.5 shows frequency spectra of vibration data of rotating machinery in different health states. Figure 2.6 shows time-frequency spectrograms of healthy and a faulty gearbox obtained by a wavelet transform and a photo of the fault.

2.2.4 Static Data

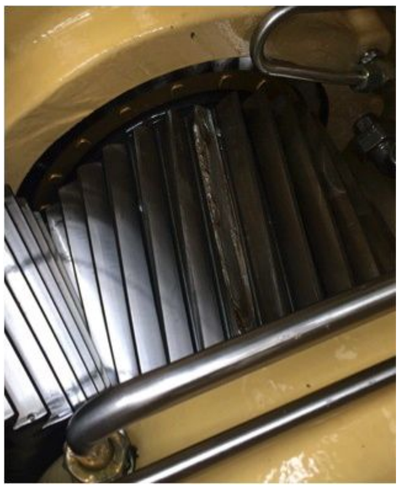
Static data include data associated with the subject that do not change over time such as installation date or model type. Installation date might be used to calculate an age of the subject — usually the higher the age the higher the probability of a failure [40]. Similarly the model type can be indicative of how likely is an occurrence of a failure [45] — e.g. a fault might frequently occur after around two years of operation for for some model types.



(a) wavelet spectrogram of a healthy gearbox

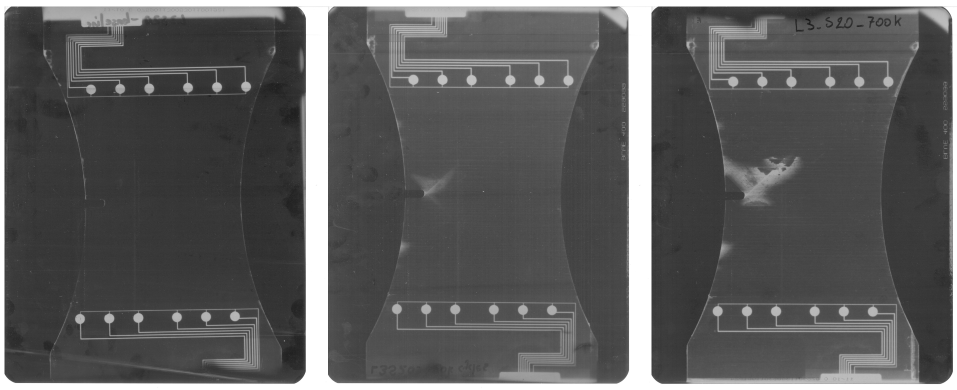


(b) wavelet spectrogram of a faulty gearbox



(c) a photo of the fault

Figure 2.6: Wavelet spectrograms of vibration data from a healthy and a faulty gearbox where the dashed vertical line separates individual rotation cycles [7] and a photo of the fault in the gearbox [8]. The faulty gearbox had a broken tooth and an increase in amplitude (darker color) once per revolution can be seen in the spectrogram.



(a) healthy (b) moderate fault (c) severe fault

Figure 2.7: X-ray images carbon fiber reinforced polymer panels degradation [9].

Events Various events such as alarms, maintenance actions or failures can happen during operation of the subject [45]. Information about past events such as number of recent alarms in past month or time from last maintenance might be indicative of how likely is the subject to fail, e.g. the longer the time from last maintenance the more likely is that it will fail in near future.

2.2.5 Health Label

Health label is a direct representation of a subject's health state. The health label can be either binary, e.g. healthy or faulty/failed, or multiclass where the different values might represent for example either a healthy state, different fault modes or different severity of faults. The health labels are typically acquired by diagnostic methods which are often performed during corrective or preventive maintenance actions [49]. In machinery, a common diagnostic method is disassembling and inspection [3] as for example shown in the Figure 2.6 which shows a fault revealed in a gearbox. The health labels might be acquired also via non intrusive methods such as X-ray imaging as shown in Figure 2.7.

2.2.6 Health Index and Failure Threshold

Health index, also called as a health indicator or a degradation level [3], represents a health state of a subject as a continuous variable. Examples of Health index are crack size, tool wear, capacity of a battery or root mean square value of vibration data. The root mean square value of vibration data is a good example of health index that is relatively easy to obtain via non-intrusive method — an accelerometer is used for measuring vibrations. However, intru-

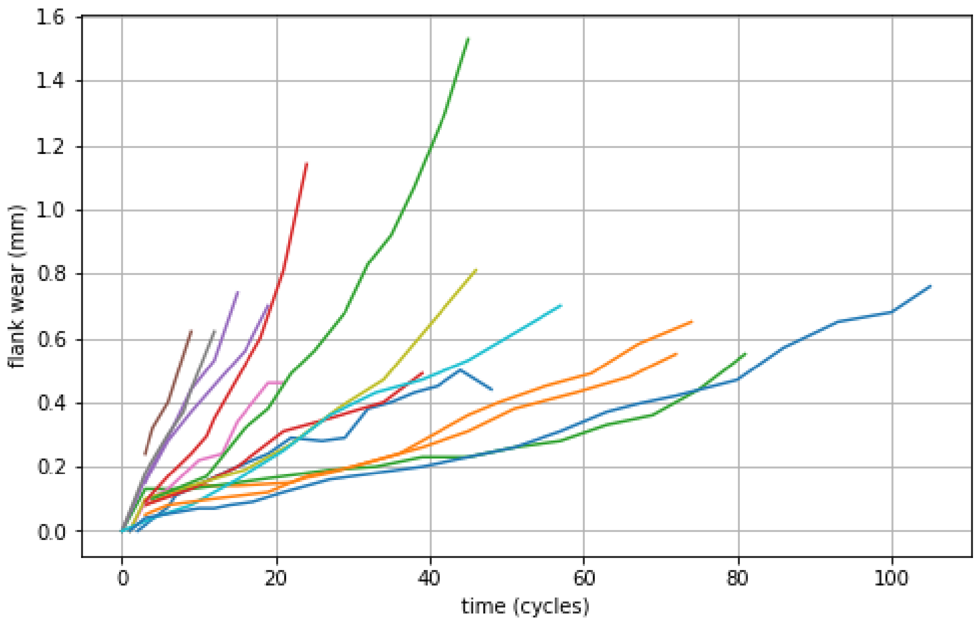


Figure 2.8: Degradation index (flank wear) of a milling machines through time. The flank wear was measured with a microscope [10].

sive methods must be sometimes used to measure the health index — Figure 2.8 shows development of a flank wear of milling machines which was measured by disassembling the milling machine and measuring the size of the wear with a use of a microscope.

A subject may be considered as failed when its health index reaches a so called failure threshold. The value of the failure threshold and can be obtained via various ways:

- by domain requirements, e.g. a minimal needed capacity of battery [50];
- by ISO norms — e.g. ISO standard 10816-3 defines permissible velocity vibration levels for machines that may be used as a failure threshold [51, 52];
- by inferring from historical data containing failures of subjects [53, 54].

2.3 Approaches to Predictive Maintenance

The goal of PdM is to monitor and analyze the condition of a subject in order to plan maintenance action when a fault is present or when a failure is likely to occur soon. In the previous section, we described the condition monitoring, i.e. the process of collecting the data that can reveal the health state of a subject. In this section we provide an introduction to approaches

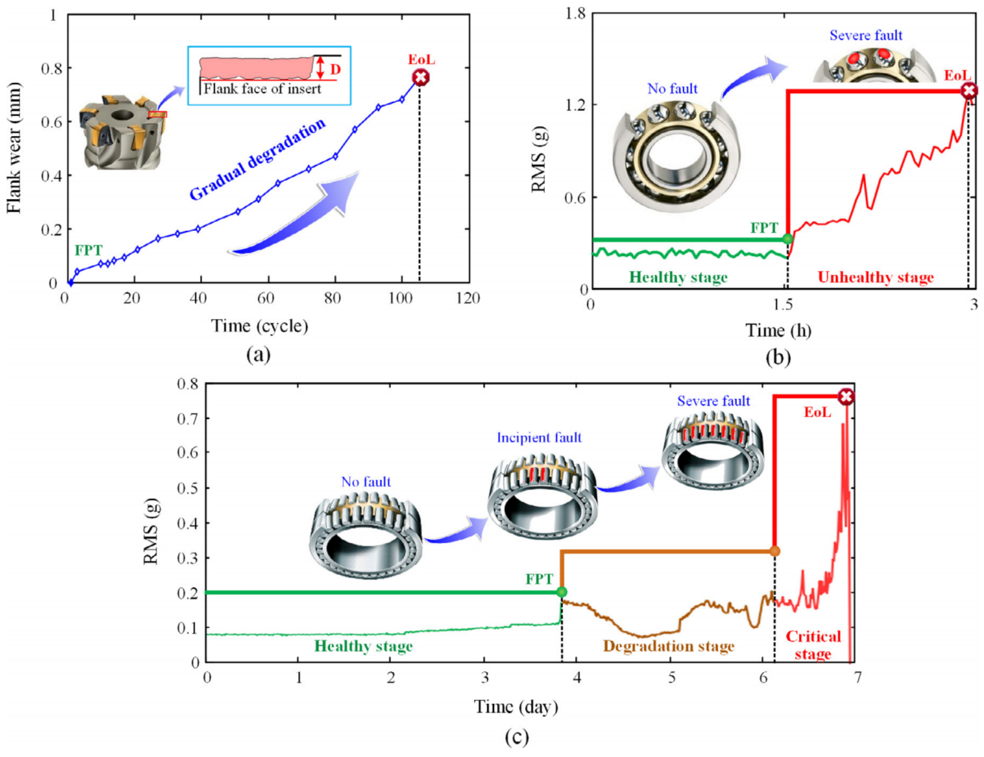


Figure 2.9: Illustration of different operational profiles of subjects [3].

how to use the condition monitoring data to build a PdM model — a model that is capable of analyzing the condition and predicting a health state of the subject.

There exist multiple typical operational profiles that might precede a failure. The common operational profiles (illustrated in Figure 2.9) include [3]:

- a continuous degradation – the subjects continuously degrades over the whole time of its operation;
- two-stage profile — a subject is healthy and operates under stable conditions until a fault occurs which starts the degradation process and potentially ends with a failure;
- multi-stage profile — similar as two-stage but the unhealthy stage can be divided into multiple stages.

Moreover, there can be available only limited data about the failures or faults — e.g. there can be only information when the subject failed, only information about the subject being faulty at some specific time point [48, 55] or there might be even no health labels available at all or of insufficient quality [17]. The existence of the different operational profiles and of the different types of available data gives rise to multiple approaches to PdM.

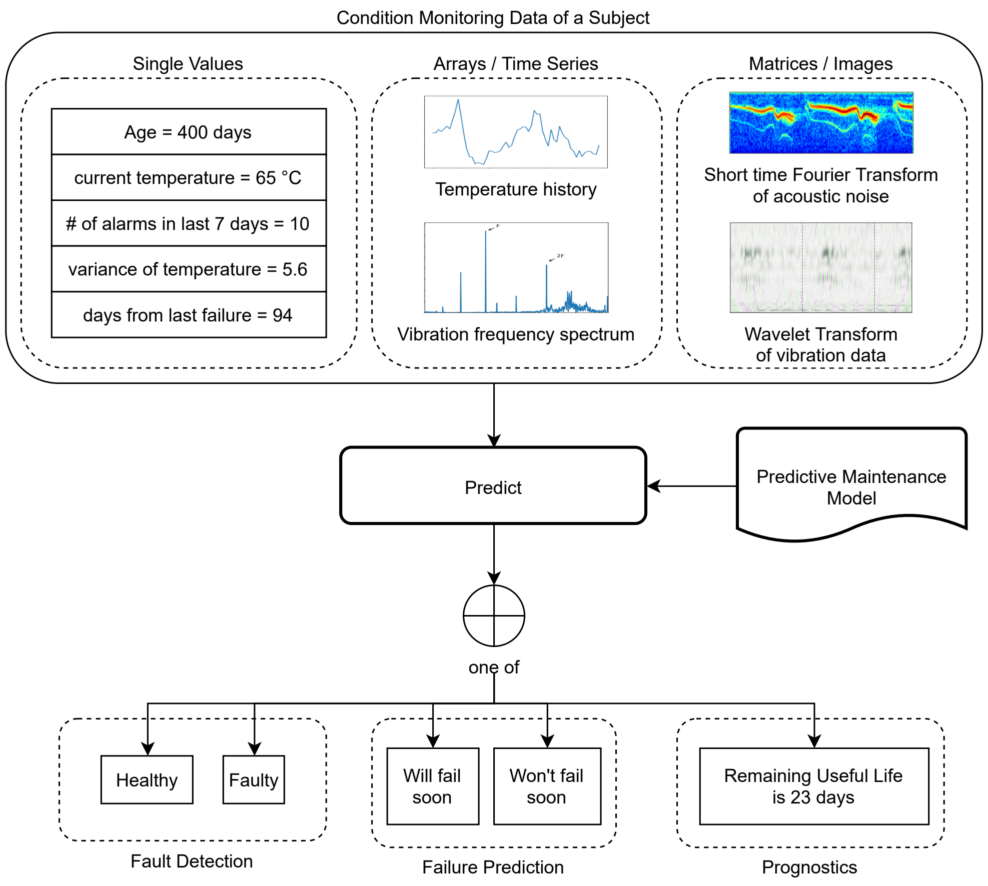

Condition Monitoring Data of a Subject
Single Values
Age = 400 days
current temperature = 65 °C
# of alarms in last 7 days = 10
variance of temperature = 5.6
days from last failure = 94
Arrays / Time Series
Temperature history
Vibration frequency spectrum
Matrices / Images
Short time Fourier Transform of acoustic noise
Wavelet Transform of vibration data
Predict
Predictive Maintenance Model
one of
Healthy
Faulty
Fault Detection
Will fail soon
Won't fail soon
Failure Prediction
Remaining Useful Life is 23 days
Prognostics

Figure 2.10: Different PdM modeling approaches from our point of view.

We identified three main approaches (illustrated in Figure 2.10):

- fault detection — detecting whether a fault (or some kind of anomaly) is present, i.e. detecting whether a subject is malfunctioning;
- failure prediction — identifying whether a failure will happen in near future;
- remaining useful life (RUL) prediction — predicting the exact amount of time that is left until a failure occurs.

The approaches differ in what operational profiles they are suitable for, e.g. in case of a continuous degradation there is no point in detecting fault but rather RUL should be predicted, and also in what data are required (e.g. a failure prediction model can be built only when there are available data about failures).

Aside the three main approaches we can see also one specific approach: fault (or failure) diagnosis — identifying which specific type of fault is present

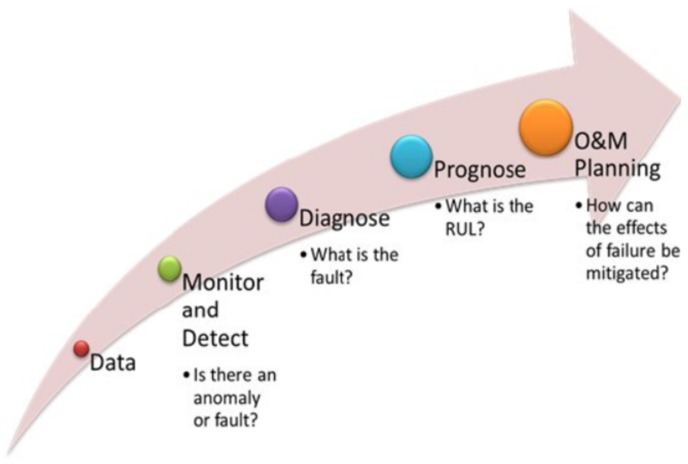


Figure 2.11: Modules in PdM according to [11].

(or which specific failure will happen) if more than one type can occur. In some literature the diagnosis is considered as part of a PdM modeling process, where the whole process consist of detecting a fault, diagnosing the exact type of the fault and making prognosis about remaining useful life [11] (illustrated in Figure 2.11). However, from the ML perspective, we consider diagnosis as a task independent of fault detection, failure prediction or remaining useful life. That is because if there exist several fault/failure a separate model can be built for each of them [56]. Therefore, we consider fault diagnosis as an extension of the approaches we describe here and we consider it as out of scope of this thesis.

In the next chapter, we describe the three main approaches mentioned above in detail.

Chapter 3

Approaches to Predictive Maintenance

The goal of PdM is to monitor and analyze condition of a subject in order to plan a maintenance action when the subject is faulty or a failure is likely to occur soon. This can be achieved by using ML algorithms and historical condition monitoring data to build a model that predicts the condition — a PdM model. In this chapter we describe three main approaches how to build a PdM model — fault detection (Section 3.1), failure prediction (Section 3.2) and remaining useful life (Section 3.3). We put an emphasis on the ML techniques used in the individual approaches and the evaluation of the built models.

3.1 Fault Detection

Fault detection is an approach where the goal is to detect whether a subject suffers from a fault or a malfunction [26]. It is thus a classification problem where the features are known condition monitoring data and the target variable is a binary health label — healthy (no fault) or faulty. When a fault is detected a maintenance action can be immediately scheduled so that a potential failure of the subject (and thus its downtime) is avoided.

From the approaches we describe in this chapter, fault detection approach is the least restrictive regarding data requirements — it does not require any data about the actual failures of the subjects.. Moreover, fault detection model can be build even when there are no health labels available at all. In that case, the faults can be considered as anomalies[3] and thus the fault detection can be formulated as an anomaly detection problem.

[3]as the fault indeed should be rare and out of distribution of the regular behaviour

Table 3.1: Example data for fault detection: (a) point-based faults, (b) range-based faults.

(a) point-based faults

features				fault
1.2	3.1	$\cdots$	4.1	0
2.1	4.2	$\cdots$	8.0	1
2.0	2.4	$\cdots$	2.2	0
1.9	1.4	$\cdots$	9.2	1
1.0	2.7	$\cdots$	2.3	0
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$

(b) range-based faults

subject id	time	features				fault
subject A	2020-01-01	0.1	0.05	$\cdots$	34.1	0
subject A	2020-01-02	0.3	0.12	$\cdots$	34.2	0
subject A	2020-01-03	1.1	3.2	$\cdots$	37.5	1
subject A	2020-01-04	1.2	3.1	$\cdots$	37.9	1
subject A	2020-01-05	0.2	0.02	$\cdots$	33.1	1
subject A	2020-01-07	2.5	0.21	$\cdots$	35.9	0
subject A	2020-01-08	2.2	0.2	$\cdots$	36.1	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$

3.1.1 Data Specifications

Fault detection approach expects condition monitoring data as the features and optionally a binary label (healthy / faulty) as the target variable. The health labels are not required as the faults can be regarded as the most anomalous samples. Based on several real-world data sets for fault detection [48, 55, 57–59] we identified two types of data for fault detection — data with range-based faults and data with point-based faults.

Range-based faults The data for fault detection can consist of time series where at each time point there is one sample that has condition monitoring data and a separate health label. The faults are thus located in time and they can last over multiple time points — consecutive samples with positive health labels (fault present) can be considered as one fault. Inspired by an article by Tatbul et al. [12] where the object of study are range-based anomalies, i.e. anomalies lasting in time, we call such faults range-based faults. Figure 3.1 shows an example of range-based faults from a real-world data set containing faults in power plants [57]. Table 3.1b then show an example of the format of the data with range-based faults.

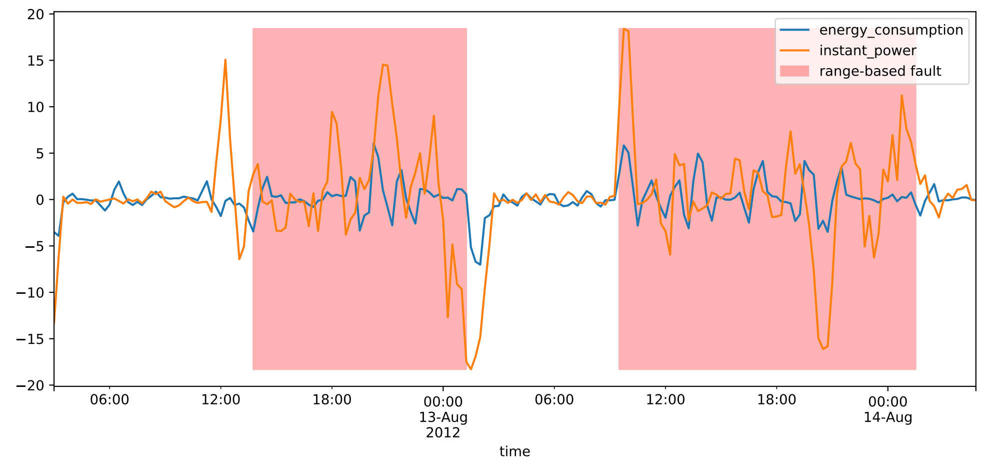


Figure 3.1: Example of range-based faults in a power plant

Point-based faults Data with point-based faults are data where each sample that contains condition monitoring data and a health label is considered as time-independent to all the other samples. Each fault can be then considered as a single point — we thus call such faults point-based. Such data set is for example a Seeded Bearing Fault Test data set from Case Western University Bearing Data Center [48] where various faults were seeded in bearings and their vibration data were measured on a test apparatus. Note, that as the vibration data are collected as signals, the data set consists of time series. However, in contrast to data with range-based fault, here each time series corresponds to one sample and thus one health label. An example of the format of the data with point-based faults is shown in Table 3.1a.

The range-based faults are commonly more realistic — in real-world the faults typically do last in time. On the other hand, the data with point-based faults can be much easier to collect — a set of healthy and faulty subjects are inspected, e.g. in a laboratory conditions or at a workshop, as for example in case of seeded bearing fault test data set mentioned above.

The range-based faults are often converted into the point-based faults before modeling as it is easier to build a fault detection model on the point-based data than on the time-series data. In the conversion, each range-based fault is split into multiple point-based ones (accordingly to the length of the range). It is important to note, though, that the range-based and point-based faults should be evaluated differently as the classical metrics for classification are not suitable for evaluation of range-based faults — they would highly favor faults with long ranges (more in Section 3.1.3).

The faults are typically rare as the subjects are most of the time healthy[4]. Therefore, real-world data sets for fault detection are commonly highly imbal-

[4]hopefully

anced with the samples having a positive label (faulty) being the minority. An exception can be data collected in laboratory conditions where for example the number of healthy and faulty samples can be the same. Such example is a condition monitoring of hydraulic systems data set [59] where multiple operation modes including a healthy mode and multiple faulty modes were simulated on a testing rig of a hydraulic system where the same amount of data was collected for every operation mode.

Another important aspect of data for fault detection is the availability of health labels. As mentioned in previous chapter (Section 2.2) the labels are typically obtained manually during e.g. corrective maintenances or by expensive methods such as disassembling of a machinery or an X-ray imaging. Therefore, it might happen that there are either no health labels available or they are not in a sufficient quantity or even quality.

3.1.2 Modeling

Fault detection is a binary classification problem — the goal is to build a model that predicts a binary class where the negative class corresponds to a healthy state and the positive class to a faulty state. The choice of the specific ML algorithm is affected by four aspects: format of the condition monitoring data (e.g. time series, spectra or simple features), type of faults (point-based faults vs range-based faults), the class imbalance and the (un)availability of the health labels.

As shown in Figure 2.10 an observation[5] of a subject can consist of a simple feature vector, one dimensional structures such as a time series or frequency spectra, images such as spectrograms or even an arbitrary combination of the mentioned. In case of a simple feature vector, classical ML algorithms such as SVM or decision trees are commonly used [60–62]. On the other hand, deep learning algorithms such as recurrent or convolutional neural networks are used as the state-of-the-art methods for fault detection with condition monitoring data containing time series or images [17, 63–65].

As the data sets for fault detection are commonly highly imbalanced (as described in 3.1.1) techniques to increase the capability of the supervised classification algorithms to classify the minority class are commonly used. Such techniques include data set balancing before the training phase or a modification of the algorithm itself [66].

In case there are only few labels available or there are no labels available at all, semi-supervised and unsupervised techniques such as anomaly detection with autoencoders can be used [17, 67].

[5]one sample in the data set that containing condition monitoring data and for which we predict the label

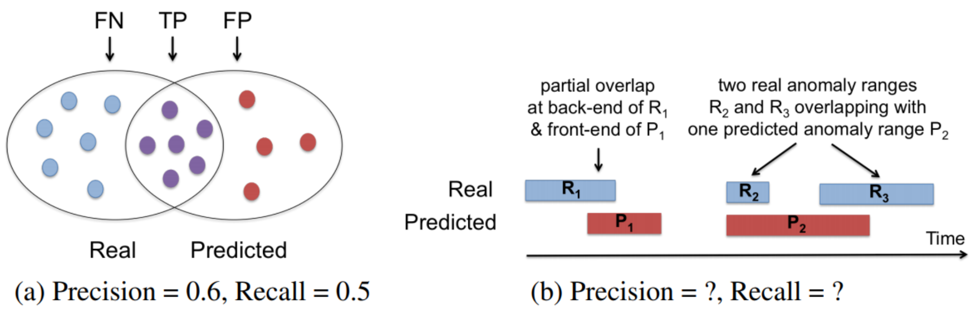

FN
TP
FP
Real
Predicted
(a) Precision = 0.6, Recall = 0.5
partial overlap at back-end of R_1 & front-end of P_1
two real anomaly ranges R_2 and R_3 overlapping with one predicted anomaly range P_2
Real
Predicted
R_1
P_1
R_2
R_3
P_2
Time
(b) Precision = ?, Recall = ?

Figure 3.2: Point-based vs range-based faults (anomalies) [12].

3.1.3 Evaluation

In this section we describe how to evaluate a performance of a fault detection model. The questions that the evaluation of a fault detection model should answer are:

- What is the probability that the model will detect a fault?
- What is the probability that the model will predict a false alarm?

The questions above are in ML commonly answered by precision and recall metrics. The evaluation of point-based faults follows classical definition of precision and recall as described in Section 1.4 as it is a standard binary classification. Regarding the range-based faults, we can convert them into point-based faults, and thus we can use the same classical evaluation metrics. However, the classical evaluation metrics might lead to misleading results for range-based faults.

In case of range-based faults the predictions are located in time, i.e. they have a start time and end time. However, the predictions are made point-wise, i.e. each time point is assigned either positive or negative label. Therefore, it might happen that a range-fault is only partially predicted (i.e. there are both positive and negative predictions during the fault). Figure 3.2 illustrates such problem where the range-based faults (in the figure named anomaly ranges) are only partially predicted. The notation used in the above mentioned figure and in the rest of this section will be as follows:

- R and R_i — the set of real fault ranges and the i^{th} real fault range, respectively;
- P and P_j — the set of predicted fault range and the j^{th} predicted fault range, respectively.

Below we define range-based recall and range-based precision metrics, for time series, respectively, as introduced by Tatbul et al. [12]. If not mentioned otherwise, all the definitions and statements below are taken and paraphrased

```
function $\omega$(AnomalyRange, OverlapSet, $\delta$)
    MyValue $\leftarrow 0$
    MaxValue $\leftarrow 0$
    AnomalyLength $\leftarrow$ length(AnomalyRange)
    for i $\leftarrow 1$, AnomalyLength do
        Bias $\leftarrow \delta$(i, AnomalyLength)
        MaxValue $\leftarrow$ MaxValue $+$ Bias
        if AnomalyRange[i] in OverlapSet then
            MyValue $\leftarrow$ MyValue $+$ Bias
    return MyValue/MaxValue
```

(a) Overlap size

```
function $\delta$(i, AnomalyLength)            ▷ Flat bias
    return 1
function $\delta$(i, AnomalyLength)    ▷ Front-end bias
    return AnomalyLength - i + 1
function $\delta$(i, AnomalyLength)    ▷ Back-end bias
    return i
function $\delta$(i, AnomalyLength)        ▷ Middle bias
    if i $\leq$ AnomalyLength/2 then
        return i
    else
        return AnomalyLength - i + 1
```

(b) Positional bias

Figure 3.3: Example definitions of an overlap size function and a positional bias function [12].

from [12]. The authors of the article define the metrics on range-based anomalies instead of range-based faults. As we use several figures from the article for illustration we stick to the term anomaly, i.e. from now on an anomaly (range) stands for a fault (range).

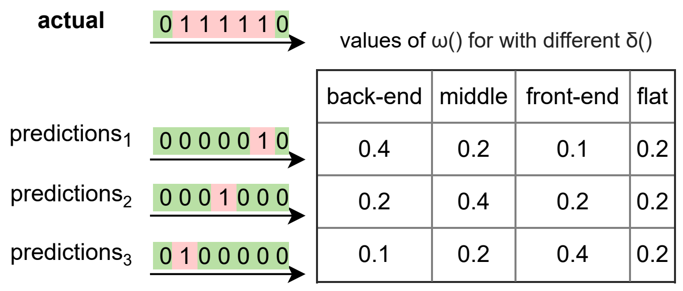
3.1.3.1 Range-based Recall

Detection of a anomaly ranges can be broken down into four aspects: existence, size, position and cardinality. We define the four aspects below and then we describe how a range-based recall can be defined with respect to these four aspects of interest.

Existence Detecting the existence of an anomaly (even by predicting only a single point in R_i) itself, might be valuable [12]. We define an existence reward function as follows:

$$\text{ExistenceReward}(R_i, P) = \begin{cases} 1, & \text{if } \sum_{j=1}^{N_p} |R_i \cap P_j| \geq 1, \\ 0, & \text{otherwise} \end{cases}$$

Size and Position The larger the size of the correctly predicted portion of R_i the better. Moreover, in some cases, not only size, but also the relative position of the correctly predicted portion of R_i might matter to the application — e.g. we might want to detect the anomaly as soon as possible. For the representation of the size and position of the overlap we use a positional bias function $\delta()$ and an overlap size function $\omega()$. The $\omega()$ function should return a value in range $[0, 1]$ where 0 is no overlap and 1 is perfect overlap (the whole real range is predicted). The $\delta()$ function is be used by the $\omega()$ function to assign weights to individual positions in the real range, i.e. $\delta()$ is a



	back-end	middle	front-end	flat
predictions$_1$	0.4	0.2	0.1	0.2
predictions$_2$	0.2	0.4	0.2	0.2
predictions$_3$	0.1	0.2	0.4	0.2

Figure 3.4: Illustration of the effect of position bias function $\delta()$ in the overlap size function $\omega()$.

parameter of $\omega()$. The simplest $\delta()$ is a flat bias — it returns the same weight for all samples. However, if for example an early prediction is more valuable, then the samples in the front of the real range can be assigned higher weight. Both of these functions can be set based on the needs of the applications. Figure 3.3 shows an example of definition of the overlap size function $\omega()$ and several examples of positional bias functions — flat, front-end and back-end. Figure 3.4 then illustrates how choice of a positional bias function $\delta()$ affects the value of $\omega()$ function. Using the $\omega()$ and $\delta()$ we define the size and the position of the overlap as

$$\sum_{j=1}^{N_p} \omega(R_i, R_i \cap P_j, \delta) \in [0,1].$$

Cardinality Detecting R_i with a single continuous prediction range $P_j \in P$ may be more valuable than doing so with multiple different predicted ranges in a fragmented manner. Therefore, we use a cardinality factor $\in (0,1]$ that expresses how many predicted ranges overlap with the real range. Cardinality factor equal to 1 is the best value, i.e. the real range overlaps with at most one predicted range, and the closer to zero the more predicted ranges overlap with it:

$$\text{CardinalityFactor}_\gamma(R_i, P) = \begin{cases} 1, & \text{if } R_i \text{ overlaps with at most one } P_j \in P \\ \gamma(R_i, P) \in (0,1], \text{otherwise.} \end{cases}$$

The value γ (overlap cardinality function) can be set for example to $1/n$ where n is the number of predicted fault ranges that overlap with the real fault range (R_i). Figure 3.5 illustrates examples of cardinality values for two different sets of predictions.

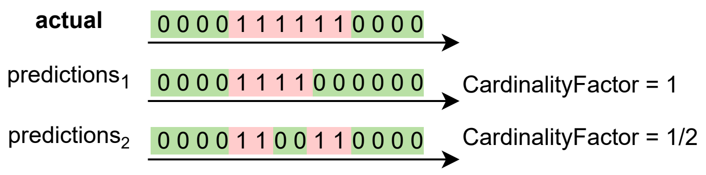


Figure 3.5: Illustration of the effect of the cardinality function ($\gamma()$) in range-based recall.

Overlap Combining the overlap size function $\omega()$ and the cardinality factor we define an overlap reward as:

$$\text{OverlapReward}_{\omega,\delta,\gamma}(R_i, P) = \text{CardinalityFactor}_{\gamma}(R_i, P) \times \sum_{j=1}^{N_p} \omega(R_i, R_i \cap P_j, \delta).$$

The overlap reward is in range $[0, 1]$ and expresses the amount of overlap including the size, position and cardinality.

Detection Score Taking the existence and overlap rewards we can quantify the amount how much each fault range is predicted by a detection score $\in [0, 1]$:

$$\begin{aligned}\text{DetectionScore}_{\alpha,\omega,\delta,\gamma}(R_i, P) =& \alpha \times \text{ExistenceReward}(R_i, P) \\ &+ (1-\alpha) \times \text{OverlapReward}_{\omega,\delta,\gamma}(R_i, P)\end{aligned}$$

where $\alpha \in [0, 1]$ is a parameter defining relative weight between the existence and the overlap reward. Setting $\alpha = 1$ represents a situation when we are only interested in whether there is at least one positive prediction in the fault range whereas $\alpha = 0$ represents a situation when we are rather interested in the amount of true predictions within the fault range.

Range-based Recall Taking the detection score we can then define the range recall as

$$\text{recall-range}_{\alpha,\omega,\delta,\gamma}(R, P) = \frac{\sum_i \text{DetectionScore}_{\alpha,\omega,\delta,\gamma}(R_i, P)}{N_r}.$$

where N_r is the amount of fault ranges.

3.1.3.2 Range-Based Precision

Range-based precision can be then defined similarly as recall with swapping real and predicted fault ranges:

$$\text{precision-range}_{\alpha,\omega,\delta,\gamma}(R, P) = \frac{\sum_i \text{DetectionScore}_{\alpha,\omega,\delta,\gamma}(P_i, R)}{N_p}.$$

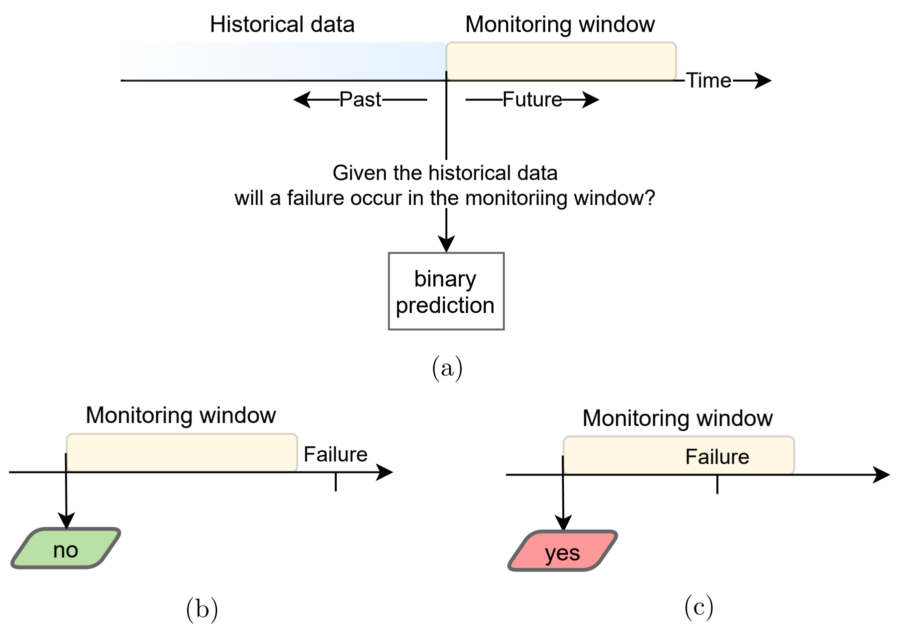


Figure 3.6: Illustration of failure prediction: (a) general concept; (b) example of negative prediction, i.e. failure won't occur; (c) example of positive prediction, i.e. failure will occur.

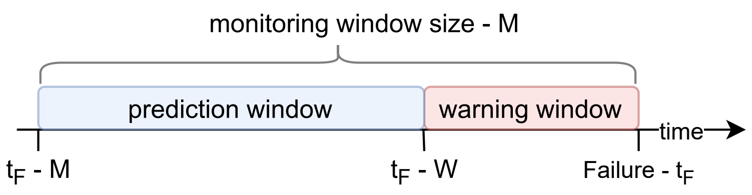


Figure 3.7: Illustration of monitoring, prediction and warning windows.

The range-based precision thus basically represents how much the predicted ranges overlap with the real ranges.

3.2 Failure Prediction

Failure prediction is an approach where the goal is to make a binary prediction whether a failure will happen in near future — in a monitoring window. The concept of failure prediction is illustrated in the Figure 3.6. Failure prediction approach is suitable in cases when there are available data about failures and when there are some patterns that precede the failure — e.g. when an air compressor raises low pressure alarms before it fails.

When predicting a failure, the domain problem might require the predictions to be made at least some time prior to the failure, e.g. the technicians

subject id	time	features				failure event
subject 1	2020-01-01	0.1	0.05	⋯	34.1	none
subject 1	2020-01-02	0.3	0.12	⋯	34.2	none
⋮	⋮	⋮	⋮	⋱	⋮	⋮
subject 1	2020-05-06	1.1	3.2	⋯	37.5	none
subject 1	2020-05-07	1.2	3.1	⋯	37.9	failure A
subject 1	2020-05-08	0.2	0.02	⋯	33.1	none
subject 1	2020-05-09	0.3	0.05	⋯	33.5	none
⋮	⋮	⋮	⋮	⋱	⋮	⋮
subject 1	2020-07-29	2.5	0.21	⋯	35.9	none
subject 1	2020-07-30	2.2	0.2	⋯	36.1	failure B
⋮	⋮	⋮	⋮	⋱	⋮	⋮

Table 3.2: A example of run-to-failure data set for failure prediction.

have to be informed at least two days ahead in order to be able to schedule and perform the maintenance. Therefore, a warning window might be specified which marks the minimal time prior to the failure in order for the prediction to be considered useful. The time period of useful predictions is then called a prediction window and is defined as the time period between $t_F - M$ and $t_F - W$, where t_F is a time stamp of the failure, M is the size of the monitoring window and W is the size of the warning window. Figure 3.7 illustrates prediction and warning windows.

Failure prediction can be seen as a special case of fault detection where the prediction window are range-based faults. However, failure prediction has some specifications in modeling and evaluation that differ from the range-based faults detection and thus we describe it as a special approach.

The concept of failure prediction is also used in many other domains such as healthcare, where heart failures are predicted, and it can also be found under name of early fault detection, early prediction of rare event, rare events prediction or rare events classification [32, 42, 43, 68]. In this thesis we will stick to the name failure prediction.

3.2.1 Data Specifications

Data for training a failure prediction model must contain information about failures. The data typically consist a condition monitoring data that are continuously collected during time and a failure log — information when failures happened. Such data are often called run-to-failure. An example of run-to-failure data is illustrated in Table 3.3. Moreover, it is necessary to have monitoring and warning windows. However, it is good to note that the monitoring time does not have to be fixed. Multiple models with different monitoring windows can be built and the choice of the final monitoring window can be made based on how the individual models perform.

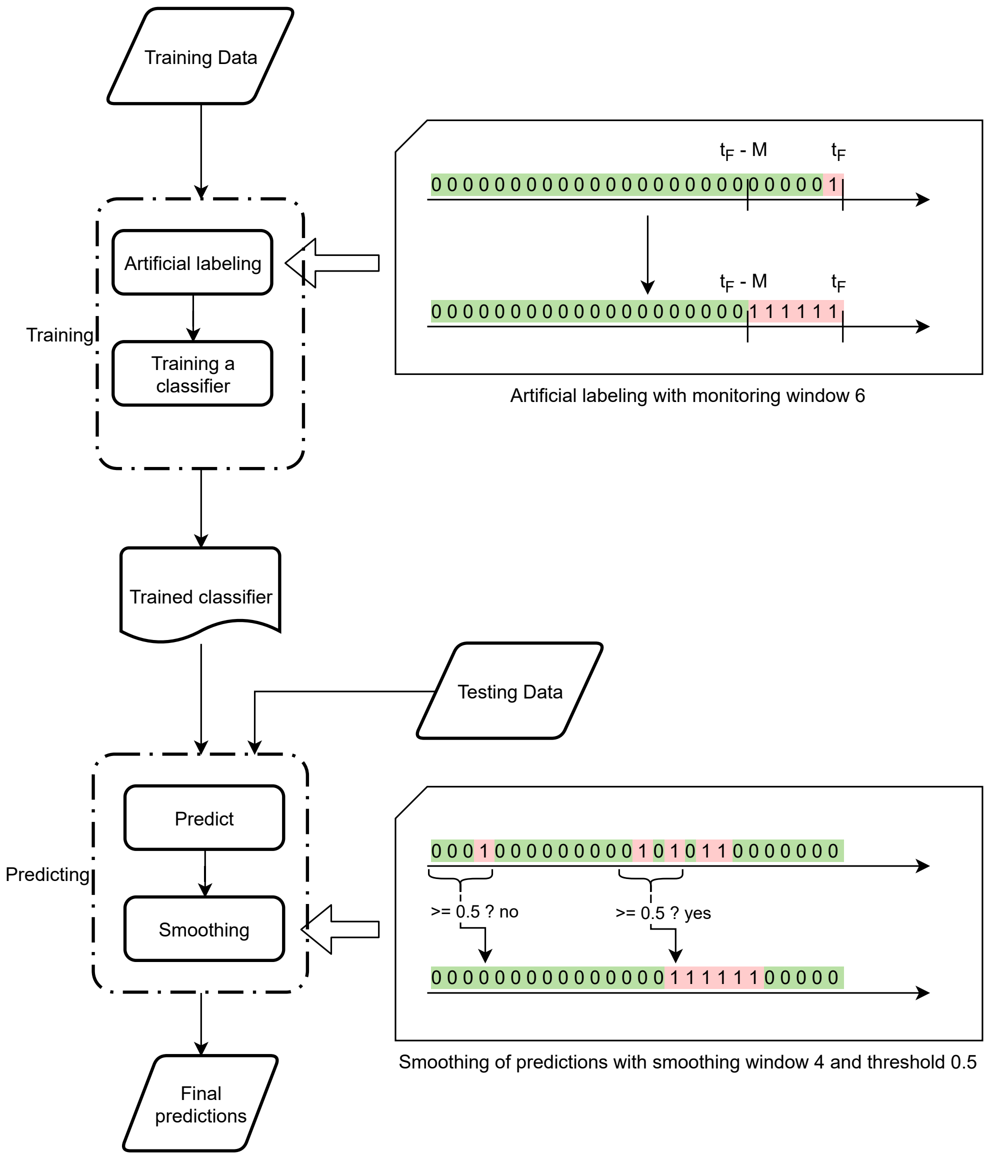


Figure 3.8: Diagram of modeling failure detection as time series point-based classification.

3.2.2 Modeling

A failure prediction model can be built using a supervised classification algorithm where the samples in the training data are artificially labelled prior to the failure as positive. Moreover, the predictions can be smoothed. The whole modeling process is visualized in Figure 3.8 and described below.

3.2.2.1 Artificial Labeling

The classifier should learn to predict positive samples prior to failures. Therefore, samples up to M time steps prior to a failure are labeled as positive. An example of artificial labeling with monitoring window of size 6 is illustrated in the upper part of Figure 3.8.

3.2.2.2 Prediction

The predictions are made point-wise by the trained classifier. However, it might happen that there occur a single positive prediction among negative predictions due to a noise. Therefore, the predictions can be smoothed using a rolling window and a positive prediction is assigned at time point t when the ratio of positive predictions in the last n predictions is higher or equal than a given threshold. We call the two parameters mentioned above a smoothing window and a smoothing threshold. In case the classifier is capable of predicting probabilities of classes, the smoothing can be performed on the probabilities before the decision threshold is applied to avoid having two thresholds.

3.2.3 Evaluation

In this section we describe how to evaluate the performance of a failure prediction model. During evaluation we will aim to answer following questions:

- What is the probability that the model will detect a failure?
- What is the probability that the model will make a false alarm? I.e. makes positive prediction but the failure won't occur in the monitoring window.

These questions are in ML commonly answered by precision and recall metrics. However, since we have more positive labels than failures due to artificial labeling, it is not straightforward how to use these metrics. Below, we describe how to use classical precision and recall (described in Section 1.4) and range-based precision and recall (described in Section 3.1.3) to evaluate a failure prediction model. Next, we describe a modification of precision and recall, reduced precision and recall, introduced by Weiss et al. [32]. Finally, we propose new precision and recall metrics, a combination of the range-based metrics and the reduced metrics, that we call event-based precision and recall.

3.2.3.1 Classical Precision and Recall

This section describes how to use precision and recall using their standard definition (Section 1.4) for failure prediction.

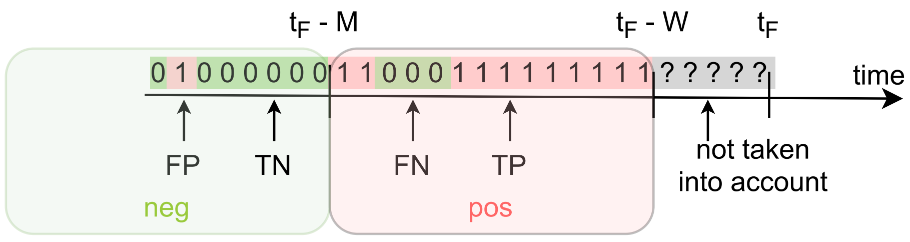


Figure 3.9: Illustration of true labels and predictions in failure prediction.

As we use artificially created positive labels, we define following:

1. Actual positive samples are samples in the prediction windows, i.e. between $t_F - M$ and $t_F - W$.
2. Actual negative samples are samples before $t_F - M$.
3. Samples in the warning window, i.e. between $t_F - W$ and t_F, are not omitted from evaluation as they do not represent useful predictions.

Figure 3.9 illustrates what time points are considered as positive and negative and which predictions are considered as TP, TN, FP and FN. Recall and precision metrics are then used by their standard definition as:

$$\text{recall} = \frac{\text{TP}}{\text{P}},$$
$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}.$$

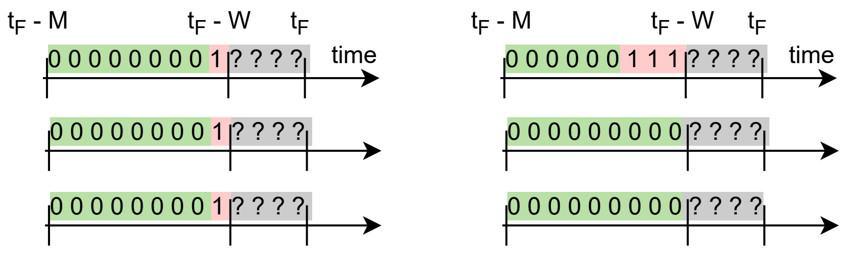


Figure 3.10: Different predictions for the same data having the same recall score: (**left**) all of the three failures predicted (**right**) only one failure predicted.

The classical metrics have two major issues:

- Recall doesn't reflect how many failures were predicted. Figure 3.10 shows examples of two models whose predictions have same recall score but they successfully predicted different number of failures.

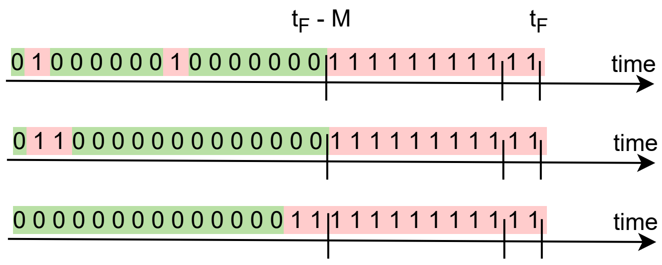


Figure 3.11: Different positions of two FPs having different severity: (**top**) far from each other — such FPs can be considered as independent; (**middle**) close to each other — the second FP is less serious and the two FPs can be almost considered as one; (**bottom**) close to each other and close to the monitoring period — probably not so serious FPs as it might happen that the failure was predicted a bit sooner than at $t_F - M$.

- Precision doesn't reflect how close FPs are to each other. Figure 3.11 shows examples of three series all having two FPs but possibly having different importance in practice, e.g. two consecutive FPs may be treated as one FP whereas two FPs far from each other should count as two FPs.

Below, we describe three other types of precision and recall metrics that might mitigate these issues.

3.2.3.2 Range-based Precision and Recall

Failure prediction can be evaluated using range-based precision and recall as described in Section 3.1.3 by taking the prediction windows as real fault (anomaly) ranges and consecutive positive predictions as predicted ranges. Setting a non-zero existence weight α to the detection score of the range-based metrics solves the first issue mentioned above — recall not reflecting how many failures were predicted.

3.2.3.3 Reduced Precision and Recall

Weiss et al. introduced in 1998 evaluation metrics called reduced precision and recall for prediction of rare events in time series [32] which is a problem identical to failure prediction. The main idea of the reduced metrics is to use the number of events (failures) instead of positive samples, use the number of predicted events instead of TP and use discounted FP instead of FP.

P $\rightarrow$ TotalEvents The total number of events (failures) is used instead of positive samples. This means that for every prediction window consisting of

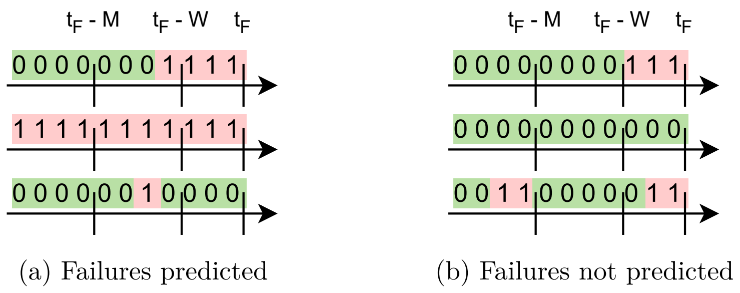


Figure 3.12: Illustrations of events (failures) being and not being predicted

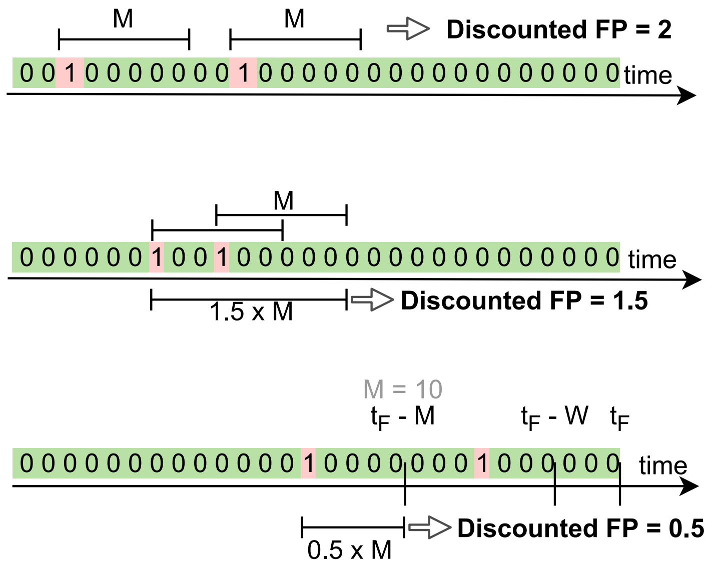


Figure 3.13: Illustration of calculating DiscountedFP.

$M - W$ samples there is one even, i.e. total number of events is equal to dividing the number of positives by the size of a prediction window.

TP $\rightarrow$ EventsPredicted The true positives are replaced by the number of predicted events. The event is considered as predicted when there is at least one positive prediction in its prediction window, i.e. at least W and at most M time steps prior to the event. Figures 3.12a and 3.12b illustrate examples of events being predicted and not being predicted, respectively.

FP $\rightarrow$ DiscountedFP Every positive prediction has a meaning that an event (failure) will happen in the monitoring window. If there are two consecutive positive predictions, the latter prediction can be considered as having lesser severity as the two consecutive predictions can be regarded as one false alarm. Therefore, Weiss et al. define calculate as the number of complete, non-overlapping, monitoring windows associated with a false positive [32]. For example when having monitoring window of size 10, two consecutive false positives will produce a discounted FP of 1.1, i.e. 1 for the first prediction plus 0.1 for the second. If the there is a false positive at time point t and a false positive at time point $t+5$, then the discounted FP for these two false positives will be equal 1.5. Figure 3.13 illustrates how discounted FP is calculated in three different scenarios.

The reduced precision and recall metrics are then defined as:

$$\begin{aligned}\text{reduced recall} &= \frac{\text{EventsPredicted}}{\text{TotalEvents}},\\ \text{reduced precision} &= \frac{\text{EventsPredicted}}{\text{EventsPredicted} + \text{DiscountedFP}}.\end{aligned}$$

Reduced recall is then identical to range-based precision when setting $\alpha = 1$, i.e. taking into account only the existence of a positive prediction in the prediction window.

3.2.3.4 Event-based Precision and Recall

Above, we described how range-based and reduced metrics can be used for failure prediction. In this section, we propose a combination of the range-based and the reduced metrics. The combination consists in taking the reduced metrics replacing the number of events predicted by a sum of detection scores from the range-based metrics. We call such metrics event-based precision and recall and define it as follows:

$$\begin{aligned}\text{event-based recall} &= \frac{\sum \text{DetectionScore}}{\text{Events}},\\ \text{event-based precision} &= \frac{\sum \text{DetectionScore}}{\sum \text{DetectionScore} + \text{DiscountedFP}}.\end{aligned}$$

where $\sum$ DetectionScore is a sum of detection scores for all the prediction windows.

3.3 Remaining Useful Life Prediction

Remaining useful life (RUL) prediction is a PdM approach where the goal is to predict the time left until the subject is still able to perform its intended function, i.e. until a failure occurs. This section is structured as follows. In Section 3.3.1 we give a motivation why and when to predict RUL instead of using fault detection or failure prediction approaches. In Section 3.3.2 we describe and compare two different modeling approaches to RUL prediction — HI-based RUL prediction and direct RUL prediction. The two approaches fundamentally differ in how RUL is predicted. However, they still provide the same output — the RUL of the subject — and thus can be evaluated the same way. Therefore, in the Section 3.3.3 we describe how to evaluate the RUL prediction independently on the chosen modeling approach.

3.3.1 Motivation

An accurate long term RUL prediction can significantly help in scheduling the maintenance actions in comparison with fault detection or failure prediction approaches. Imagine a situation of having a large amount of subjects which started operating at the same time — for example a fleet of one hundred wind turbines. If the turbines were operating under similar conditions it might happen that they will all tend to fail after similar amount of time of operation — e.g. after two years. Both fault detection and failure prediction approaches would then notify that all the wind turbines are faulty (or are going to fail soon) at a similar time shortly before the failures, let's say one week ahead. However it might be impossible to schedule maintenance actions for all the wind turbines at that time point because only two wind turbines can be maintained per day and there is one hundred of wind turbines about to fail in one week. With an accurate RUL prediction, on the other hand, one can continuously have information about when each individual subject is going to fail. If the RUL is then similar for many subjects the maintenance actions can be scheduled more in advance so that all the subjects are maintained in advance. However, an accurate RUL prediction is typically possible only in certain domains and only when having the right type of data.

RUL prediction is typically done on subjects which have an ongoing continuous degradation that can be well quantified — for example a turbine bearing deterioration. The failure then can be either a complete inability of the subject to operate (e.g. the wind turbine shuts down) or it can be a state when the subject is no longer capable of safe operation or of operation at enough quality — e.g. a maximum capacity of a battery reaches 40 % of the designed capacity[6]. In such cases it is common to predict RUL during the whole lifetime of the subject (e.g. every day or week) [15]. In other cases, however, the subject might operate under stable, healthy, conditions without any sings of wear until a fault occurs which triggers the degradation process — the fault grows in severity (as illustrated in the beginning of this chapter in Figure 2.9). In such cases the RUL can be predicted over the whole lifetime as well but it might happen that the prediction would be highly inaccurate until the fault occurs. Therefore, the RUL prediction can start after the fault (or anomaly) is detected[7] [3] and thus providing an estimation of the fault's severity.

[6]In such cases the failure is often rather called an end of life (EoL). However, as it principally represents the same thing we will stick to the naming convention of failure.

[7]assuming the fault is detected early enough so that the RUL prediction is useful

subject id	time	features			failure	**RUL**
subject A	2020-01-01	0.1	$\cdots$	34.1	0	127
subject A	2020-01-02	0.3	$\cdots$	34.2	0	126
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$
subject A	2020-05-05	1.1	$\cdots$	37.5	0	2
subject A	2020-05-06	1.1	$\cdots$	37.5	0	1
subject A	2020-05-07	1.2	$\cdots$	37.9	1	0
subject B	2020-01-01	0.2	$\cdots$	33.1	0	89
subject B	2020-01-02	0.3	$\cdots$	33.5	0	88
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$
subject B	2020-03-29	2.5	$\cdots$	35.9	0	1
subject B	2020-03-30	2.2	$\cdots$	36.1	1	0

Table 3.3: Example of calculating the RUL values from the run-to-failure data.

3.3.2 Approaches

We identified two different modeling approaches to RUL prediction — direct RUL prediction and HI-based RUL prediction[8]. In this section we describe and compare the two approaches.

3.3.2.1 Direct RUL Prediction

Direct RUL prediction is suitable when there are available run-to-failure data for at least several subjects. The approach consists in training a regression model on the run-to-failure data where the regressands (features) can be any known data about the subject at the time of the prediction and the regressor (the predicted value) is the RUL retrospectively calculated from the run-to-failure data. The calculation of the RUL is typically done as follows — at the time of the failure, T, the RUL is equal to 0, at time $T-1$ the RUL is equal to 1, at $T-2$ the RUL is equal to 2 and so on. Table 3.3 shows an example of run-to-failure data set and calculating of the RUL.

As described in the previous section, the subjects might operate under stable and healthy conditions until a fault occurs which triggers the degradation process. In that case the RUL prediction might be very inaccurate at the early stage of the subject's operation where it might be hard to distinguish between e.g. RUL of 200 days and 170 days. Therefore in some works limiting of the RUL values with an upper bound is suggested [13] — e.g. all the RUL values above 130 are set to 130 see Figure 3.14 for illustration). The authors in [13] conclude that such clipping might result in better performance in terms of root mean squared error metric. However, as will be discussed in Section 3.3.3 the RMSE metric might be unsuitable for evaluation of RUL prediction

[8]HI — health indicator

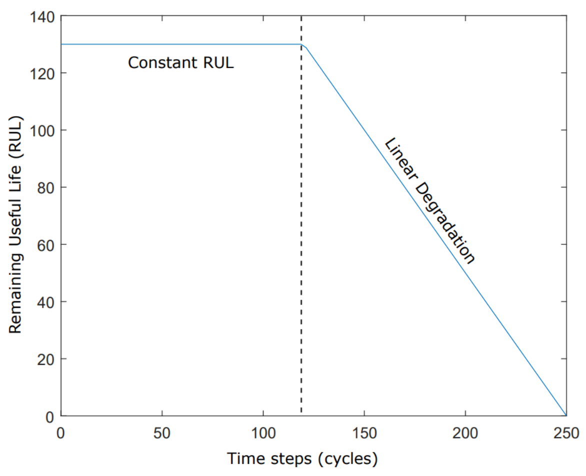


Figure 3.14: Illustration of limiting RUL with an upper bound [13].

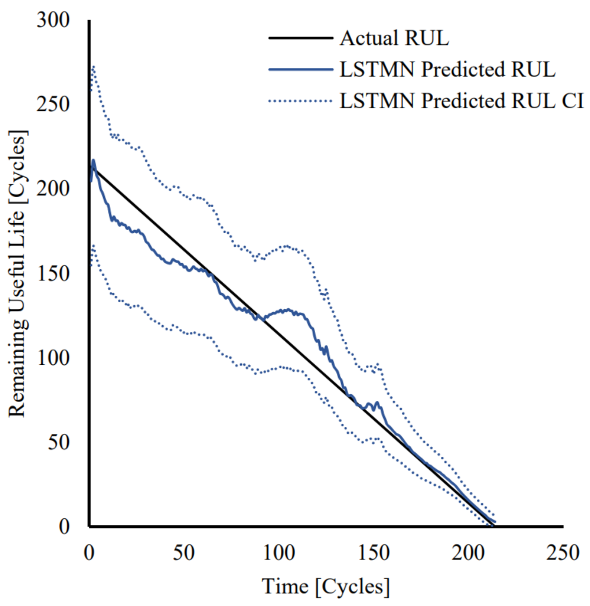


Figure 3.15: Illustration of RUL prediction with a Bayesian LSTM neural network [14].

performance and thus it is questionable whether this RUL clipping helps the overall performance of the model.

Promising results of direct RUL prediction have been achieved with recurrent and convolutional neural networks in various domains such as wind turbines or bearings [69, 70]. In recent years, Bayesian neural networks are gaining on interest in direct RUL prediction as they can predict a probability density function (PDF) instead of single value predictions [14, 71]. The mean

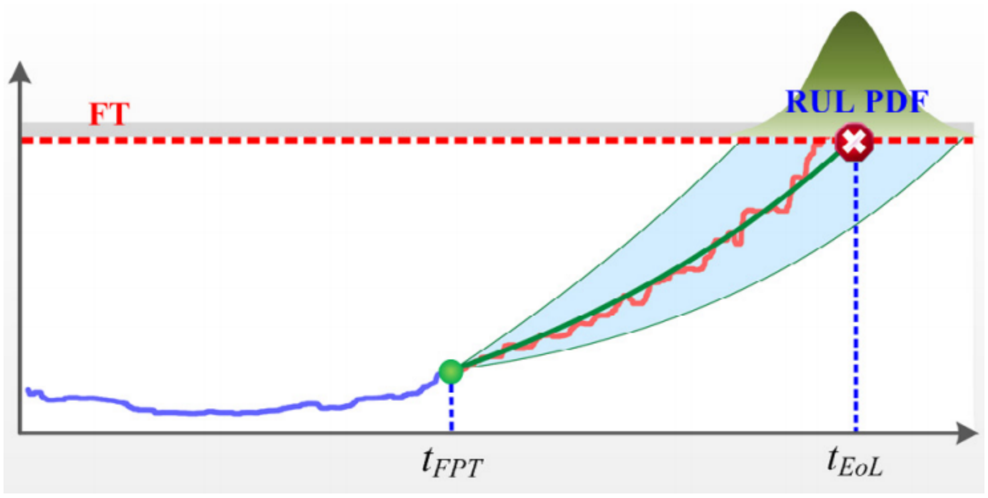


Figure 3.16: Illustration of HI-based RUL prediction. The red dashed line represents a failure threshold (FT), the blue line represents a health indicator up to a current time point (green dot), the green line shows a prediction of the health indicator in the future and the red line represents the actual future values of the health indicator. [3].

of the PDF can then be used as the predicted RUL and a confidence interval can be calculated and used as a form of uncertainty — which might be very valuable for the end users as a supportive information about the prediction. Figure 3.15 illustrates prediction of RUL with a Bayesian LSTM neural network.

In some literature, the direct RUL prediction approach is considered as having relatively low capability of predicting the RUL since a linear relationship between the RUL and the condition monitoring data is established [26].

3.3.2.2 HI-based RUL Prediction

The HI-based RUL prediction approach is suitable in cases when there is available a health indicator (HI) that directly represents the subject's health state and a predefined failure threshold. The failure of the subject is then considered as the time point when the HI crosses the failure threshold. A typical example of such case are batteries where the health indicator can be their current maximal capacity and the failure threshold can be a ratio of the designed maximal capacity (e.g. 30 %). The RUL prediction then consists in building a model that forecasts future values of the HI and in identifying the time point when the HI crosses the failure threshold. The RUL is then calculated as the time difference between the identified time point of the crossing and the current time point.

The forecasted HI is commonly in a form of a probability density function

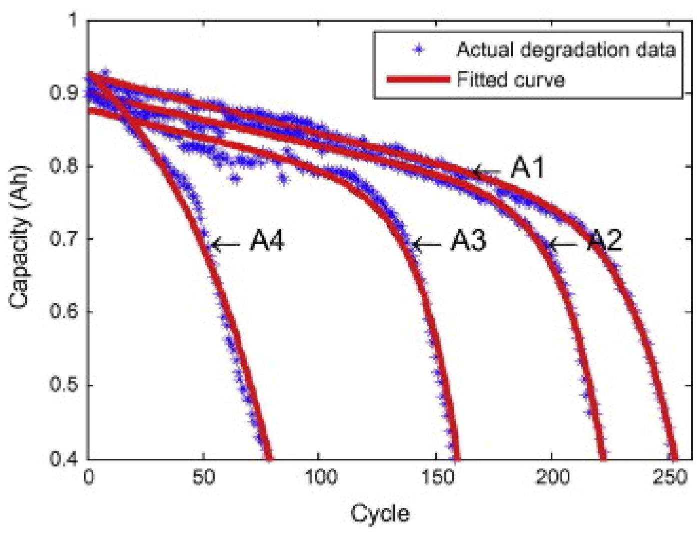


Figure 3.17: Finding an empirical model for degradation of battery capacity [15].

(PDF) which tends to have higher variance the farther to the future the HI is predicted [30]. In case of predicting the PDF of the HI the failure can be defined as the time point when the mean of the PDF crosses the failure threshold. Figure 3.16 illustrates the forecasting of the HI with a PDF.

The HI forecasting techniques can be divided into two categories:

- model-based techniques;
- machine learning techniques;

Model-based Techniques The model-based techniques are based on the existence of an underlying physical or statistical model that describes the degradation process of the subject. Such physical or statistical models can be either apriori known or empirically observed from the available data. For example the capacity of the batteries can be commonly fitted by an exponential model [72] — see Figure 3.17 for illustration. The physical and statistical models have typically parameters which are estimated from the currently available HI data of the subject (the parameters are estimated for each subject separately). In other words the forecasting at time point t is done by a model with parameters estimated based on the HI data up to time point t.

ML Techniques The AI techniques, on the other hand, do not require any domain knowledge about the degradation process but rather build a model that learns the degradation patterns from the previous HI data points by itself. A simple technique can be using a regression model that takes time as the regressand and the HI as the regressor [73]. More advanced techniques include for example a recurrent neural networks which can be trained to time series prediction, i.e. output future values of the HI based on previous values [74].

A big benefit of HI-based RUL prediction over the direct RUL prediction is that there is no need for run-to-failure data. The unavailability of run-to-failure data is common in many domains such as aviation where the subjects simply cannot operate until a failure mostly from safety reasons [30]. On the other hand, the downside of the HI-based RUL prediction is that there has to be a well defined health indicator and failure threshold.

3.3.3 Evaluation

In this section, we describe how to evaluate the performance of a RUL prediction model. We will use the following notation:

- N — number of subjects;
- $n \in [1, N]$ — index of a n-th subject;
- $t \in [1, T_n]$ — time index of n-th subject's observations, where $t = 1$ is the first observation and $t = T_n$ is the time stamp of the failure;
- $\mathrm{RUL}_n(t)$ — actual RUL of n-th subject at time t;
- $\widehat{\mathrm{RUL}}_n(t)$ — predicted RUL of n-th subject at time t;
- W — warning time;
- $\mathrm{EoUP}_n = T_n - W$ — end of useful predictions (EoUP), the last time stamp of useful predictions, i.e. not yet in the warning window.

The warning time has the same meaning as in failure prediction — some lead time before the failure of the subject might be necessary so that the maintenance action can be scheduled and performed. Therefore, we will exclude the time points between $T - W$ and T from the evaluation.

3.3.3.1 Classical Metrics

RUL prediction is in context of ML a regression task — the goal is to predict a continuous variable. Thus, classical regression metrics such as MAE, RMSE or MAPE can be used. However, it is not straightforward how to use them from two reasons. First, we should omit the predictions made in the warning window, the predictions after EoUP. Second, we can calculate the mean error either over subjects or over individual samples.

An absolute error, squared error and absolute percentage error for n-th subject can be calculated as:

$$\begin{aligned}
\text{AbsoluteError}_n &= \sum_{t=1}^{\text{EoUP}_n} \left|\text{RUL}_n(t) - \widehat{\text{RUL}}_n(t)\right|, \\
\text{SquaredError}_n &= \sum_{t=1}^{\text{EoUP}_n} (\text{RUL}_n(t) - \widehat{\text{RUL}}_n(t))^2, \\
\text{AbsolutePercentageError}_n &= \sum_{t=1}^{\text{EoUP}_n} \left|\frac{\text{RUL}_n(t) - \widehat{\text{RUL}}_n(t)}{\text{RUL}_n(t)}\right|.
\end{aligned}$$

In general, we will denote the error of the n-th subject as Error_n. The mean of the errors can be then calculated either over subjects or over samples:

$$\begin{aligned}
\text{mean error over subjects} &= \frac{1}{N}\sum_{n=1}^{N} \frac{\text{Error}_n}{\text{EoUP}_n}, \\
\text{mean error over samples} &= \frac{1}{\sum_{n=1}^{N} \text{EoUP}_n} \sum_{n=1}^{N} \text{Error}_n.
\end{aligned}$$

Mean absolute percentage error (MAPE) over samples can be then calculated as:

$$\text{MAPE} = \frac{1}{\sum_{n=1}^{N} \text{EoUP}_n} \sum_{n=1}^{N} \sum_{t=1}^{\text{EoUP}_n} \left|\frac{\text{RUL}_n(t) - \widehat{\text{RUL}}_n(t)}{\text{RUL}_n(t)}\right|$$

The metrics such as MAE and RMSE are then calculated analogically.

3.3.3.2 Prognostic Horizon

Prognostic horizon is an evaluation metric specifically tailored for RUL prediction proposed by Saxena et al. [30]. It aims to answer the following question: What time ahead of the failure are the predictions within a prespecified bound around the actual RUL for one specific subject? The prognostic horizon is defined as the time difference between the time of failure (T_n) and the first time index from which all the future predictions are within the boundaries specified by parameter α. For the n-th subject the prognostic horizon (PH) with parameter α is defined as:

$$\text{PH}_{\alpha,n} = T - \min\left\{i | \forall t, t \geq i : \widehat{\text{RUL}}_n(t) \in [\text{RUL}_n(t) - \alpha, \text{RUL}_n(t) + \alpha]\right\}.$$

Figure 3.18 illustrates the calculation of prognostic horizon on predictions for one subject.

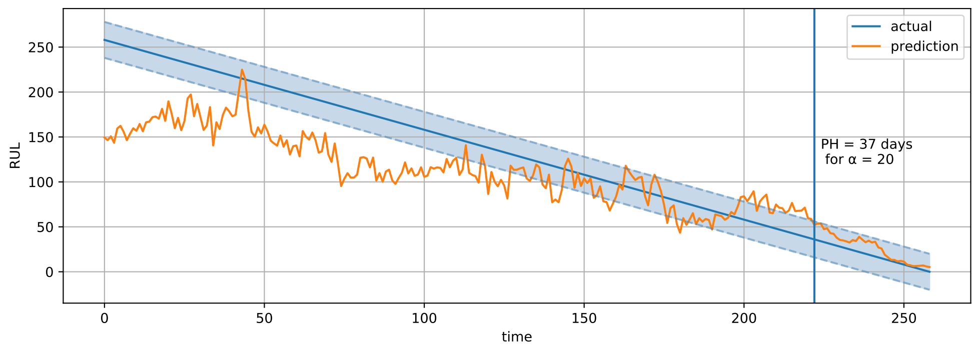


Figure 3.18: Illustration of prognostic horizon.

For evaluation of multiple subjects, mean prognostic horizon $(\mathrm{MPH})_\alpha$ can be used as:

$$\mathrm{MPH}_\alpha = \frac{1}{N}\sum_n \mathrm{PH}_{\alpha,n}$$

3.3.3.3 Metrics Relative to RUL

It might happen that the RUL prediction model has relatively high errors at high RUL values compared to the errors at low RUL values. Therefore, we can calculate the errors only for RUL values lower than a certain fixed value, for example calculate MAPE only for RUL values lesser than 40. We define a mean absolute percentage error up to RUL values equal to k (MAPE@k) as:

$$\mathrm{MAPE@k} = \frac{1}{\sum_{n=1}^{N}(\mathrm{EoUP}_n - k - 1)}\sum_{n=1}^{N}\sum_{t=k}^{\mathrm{EoUP}_n}\left|\frac{\mathrm{RUL}_n(t) - \widehat{\mathrm{RUL}}_n(t)}{\mathrm{RUL}_n(t)}\right|$$

Metrics such as MAE@k, RMSE@k can be then defined analogically.

In RUL prediction, we might be interested in how the model performs at relatively to the actual RUL values, i.e. we might want to obtain one score for every actual RUL value. Therefore, instead of calculating a mean score over all the samples, we can calculate a mean score over all subjects at a various RUL values. MAPE at different RUL values is defined as follows:

$$\mathrm{MAPE(RUL)} = \sum_n \frac{\mathrm{RUL} - \widehat{\mathrm{RUL}}_n(T_n - \mathrm{RUL})}{\mathrm{RUL}}.$$

The metrics MAE, RMSE and others can be then defined analogically.

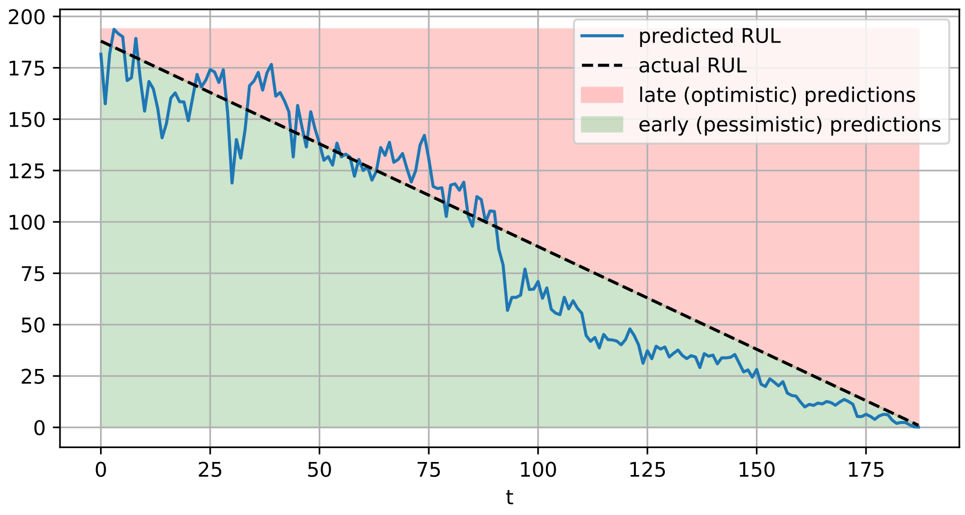


Figure 3.19: Illustration of optimistic and pessimistic predictions.

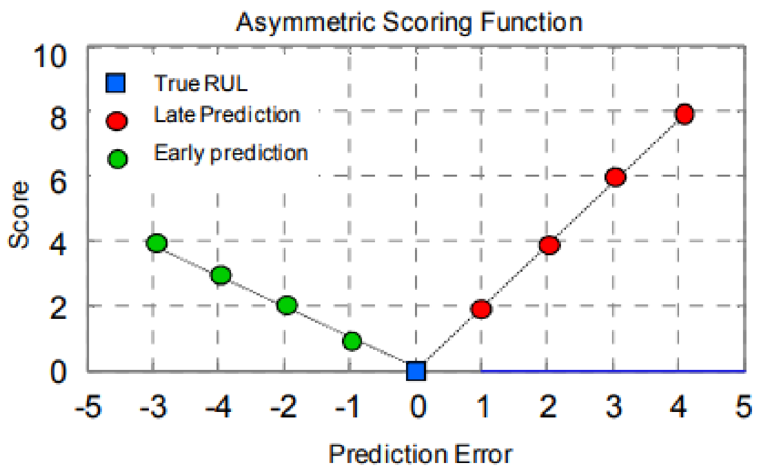


Figure 3.20: Asymmetric weighting function for late and early predictions [16].

3.3.3.4 Asymmetric Weighting

When the prediction is not perfect, i.e. RUL $\neq \widehat{\text{RUL}}$, it can be either late (optimistic) if RUL $> \widehat{\text{RUL}}$ or early (pessimistic) if RUL $< \widehat{\text{RUL}}$. Figure 3.19 illustrates RUL predictions for one subject and highlights what predictions are late and which are early. The late predictions might bring a risk that the subject fails before the maintenance is performed. Therefore, the early predictions are typically better since the subject is in the worst case maintained earlier than it would have to be.

In [75] and [16] the authors suggest using an asymmetric weighting function for weighting the prediction errors. The main idea is to give higher weight to the late predictions in comparison with the early ones. The asymmetric weighting function φ can be for example defined as

$$\varphi(d) = \begin{cases} -d & \text{if } d < 0 \\ 2d & \text{if } d \geq 0 \end{cases}$$

where d is the difference between the predicted RUL and actual RUL, i.e. $d = \widehat{\mathrm{RUL}} - \mathrm{RUL}$. Figure 3.20 illustrates the above defined asymmetric weighting function.

The weighting function φ can be then used in the classical regression metrics. We define a mean asymmetrically weighted percentage error (MAWPE) with an asymmetric scoring function φ as:

$$\mathrm{MAWPE}_\varphi = \frac{1}{\sum_{n=1}^{N} \mathrm{EoUP}_n} \sum_{n=1}^{N} \sum_{t=1}^{\mathrm{EoUP}_n} \left| \varphi \left(\frac{\mathrm{RUL}_n(t) - \widehat{\mathrm{RUL}}_n(t)}{\mathrm{RUL}_n(t)} \right) \right|$$

The usage of the asymmetric weighting function in other metrics is then analogical.

The asymmetry can be also applied in prognostic horizon where instead of a single parameter α (defining the width of the bound) two parameters α^+ and α^- can be used which define the lower and the upper boundary, respectively.

Chapter 4

Experiments

In this chapter we describe experiments conducted on real-world publicly available PdM data sets. There are two main goals of the experiments:

- demonstrate the approaches to PdM utilizing ML techniques;
- compare the different evaluation metrics that can be used within each approach.

For each of the approaches we conduct one experiment — totaling in three experiments.

Though there is not enough publicly available data sets to make a robust comparison of the metrics, we can conduct each experiment on one carefully selected data set, suitable for the given approach, and compare the metrics from two aspects:

- model selection — Do evaluation metrics differ in how they rank different models for the task related to the data set?
- interpretability and practical value — How can the evaluation metrics be interpreted by a domain expert? Does the evaluation bring practical value for the task related to the data set? For example do the metrics clearly express what the probability of predicting a failure or detecting a fault is?

As the approaches and the evaluation metrics are different, every experiment has slightly different experiment design. The general design of the experiments can be, however, summarized as follows:

1. candidate models selection — use multiple evaluation metrics to select a set of candidate models from a larger amount of built models, e.g. select best ranked model by every metric, and discuss how the models agree in models' ranking;

2. candidate models comparison — compare the candidate models, e.g. using PR analysis, and discuss which model might be the most suitable model for the given task;

3. performance interpretability — discuss what will the model's performance in practice be.

This chapter is organized as follows. In Section 4.1 we describe implementation of the experiments including chosen technologies and the experiments' reproducibility. In Sections 4.2, 4.3 and 4.4 we describe the three conducted experiments, respectively. Each experiment consists of sections:

- data set description,
- task definition,
- design of experiment,
- results and
- discussion.

4.1 Implementation

The full implementation of the experiments is available as a GitHub repository[9]. In this section, we provide a brief overview about the implementation details.

4.1.1 Technologies

We implemented the experiments as Jupyter notebooks [76] in Python using standard Python libraries for machine learning, scientific computations, and visualizations including Scikit-learn [77], NumPy [78], SciPy [79], Matplotlib [80], Seaborn [81], XGBoost [36] and Pandas [82]. For the calculation of range-based precision and recall metrics we used a python implementation the authors of [12] provide at GitHub [83]. Moreover, for the calculation of AUPRG we used pyprg package [84].

4.1.2 Hardware

We ran the experiments on a computational cluster using 256 GB of RAM and 32 CPU cores of AMD Opteron 6344 CPU units provided by Datamole[10].

[9] `https://github.com/datamole-ai/predictivemaintenancethesis`
[10] `www.datamole.cz`

4.1.3 Reproducibility

Each experiment is completely reproducible. We use Poetry package manager [85] to ensure that the same versions of Python libraries we used can be installed for reproducing the experiments. Our GitHub repository thus contains a `pyproject.toml` and `poetry.lock` files which can be used to install all the Python packages at the same version we used. We used a fixed random seed in all parts of code that depend on randomness so that reproducing the experiments always yields the same results. Reproducing each experiment then consist in running a Jupyter notebook which contain code for downloading the publicly available data set, preprocessing of the data and all the other steps of the experiments such as modeling and visualizing the results.

4.2 Experiment — Fault Detection in Scania Trucks

Fault detection is an approach where the goal is to build a model that detects faulty behaviour, malfunction, of the subject. It can be modeled as a binary classification or an anomaly detection — depending on whether health labels are available or not. It is suitable in cases when there are no or insufficient data about actual failures, i.e. breakdowns, of the subjects. In this section we describe an experiment where we demonstrate this approach and compare its evaluation metrics on one real-world data set.

The data used for building a fault detection model can contain either point-based or range-based faults, i.e. faults with no temporal location or faults located in time and lasting for a certain period of time (for more details see Section 3.1.1). Since detection of range-based faults is partly similar to failure prediction approach, which we demonstrate in the following experiment, we focus in this experiment on fault detection of point-based faults.

For the purpose of our experiment we choose a data set containing point-based faults in air pressurized system of Scania trucks [49]. We choose this data set because its authors clearly define an objective function — a cost function assigning costs to false alarms and missed faults (FPs and FNs) expressed in the amount of dollars. The authors then clearly define the task as to build a model that minimize this cost function. The cost function allows us to demonstrate how decision threshold of the built classifier can be selected in practice.

4.2.1 Data Set Description

The data set we use in this experiment contains condition monitoring data and point-based faults in an air pressure system of heavy Scania trucks [49]. It consists of 76000 records and 171 columns where each row represents one truck with the first column containing a binary health label (positive class represents a fault) and the next 170 columns containing anonymized features. The provided data are already split into training and testing set where the training set contains 60000 records and the testing set contains 16000 records. Both the sets are highly imbalanced — the ratio of positive classes to the total amount of records is approximately 1.67 % in the training set and 2.34 % in the testing set.

Preprocessing The data set is provided partially preprocessed in a form of two CSV files, one for training set and one for testing set. All the features are numerical of which some of them are provided already binned — i.e. split to a finite number of intervals. There are some missing values in multiple feature columns which we impute by a mean of each column in the training data[11]. Otherwise, we consider the data set as preprocessed and suitable for the classification algorithm we chose (described later below).

4.2.2 Task Definition

The authors of the data set define a cost function assigning costs in dollars for the false predictions:

- $ 10 per FP — cost of unnecessary check needed to be done by a mechanic at workshop;
- $ 500 per FN — missing a faulty truck that may cause a breakdown.

The task the authors set is to thus build a fault detection model using the training set that predicts whether a fault is present in the truck and to minimize total cost of the false predictions in the testing set.

4.2.3 Design of Experiment

Our data set contains enough labels for both positive and negative samples and thus we can approach the fault detection as a supervised classification. For the purpose of our experiment, we choose only one type of classification algorithm — gradient boosted trees, more specifically its Python implementation XGBoost [36]. XGBoost has several hyperparameters that should be tuned such as number of trees (estimators), max depth of the trees or minimal number of samples to perform a further split (min_child_weight). Moreover, the XGBoost is a probabilistic classifier, i.e. it can predict probabilities instead of the binary classes themselves. A decision threshold that optimizes minimizes the total costs can be thus tuned.

[11]Note that it is very important not to impute missing values by a mean values of the whole data set as that would contaminate the data training data with the information from the testing data.

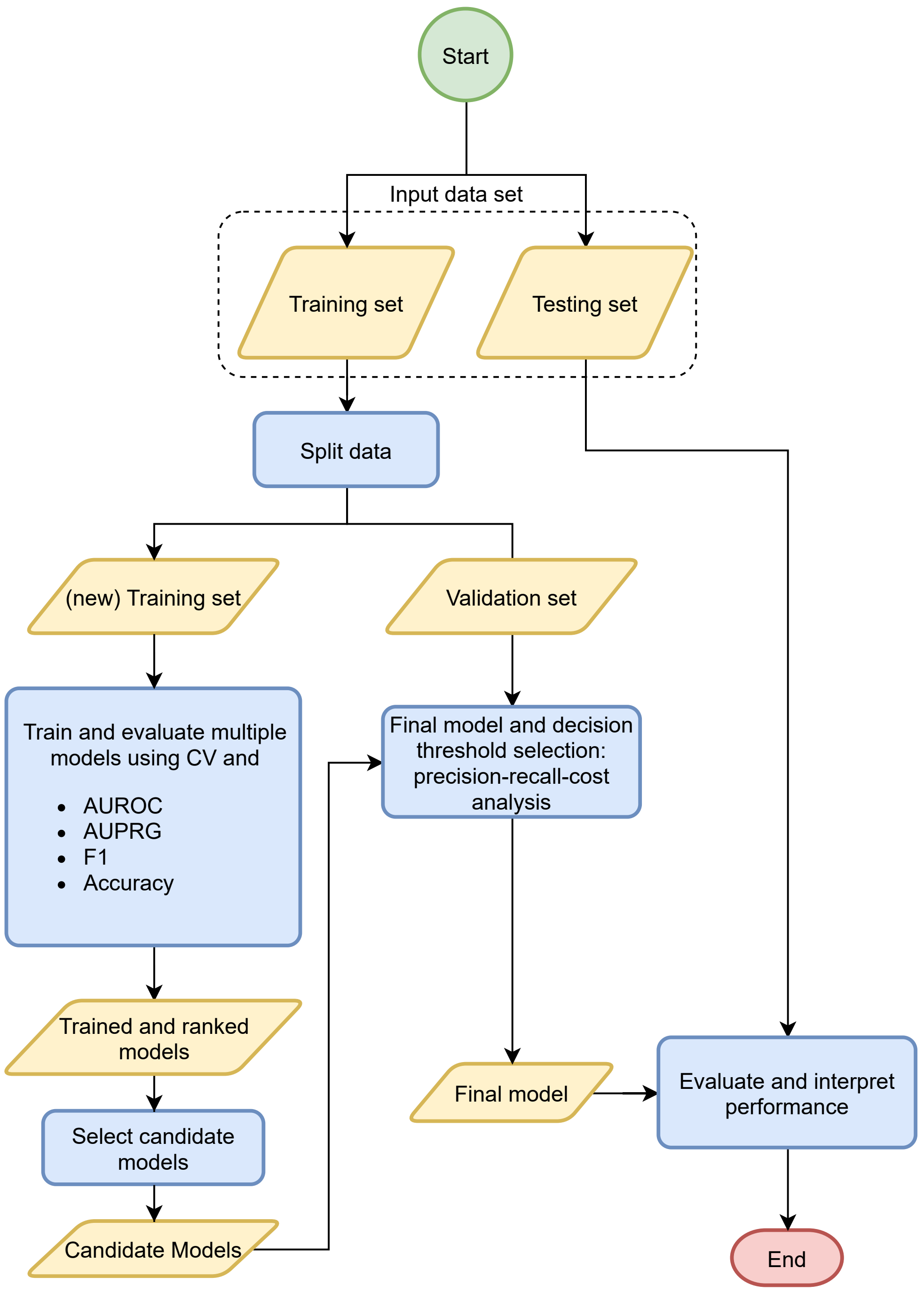

Start
Input data set
Training set
Testing set
Split data
(new) Training set
Validation set
Train and evaluate multiple models using CV and
• AUROC
• AUPRG
• F1
• Accuracy
Final model and decision threshold selection: precision-recall-cost analysis
Trained and ranked models
Select candidate models
Candidate Models
Final model
Evaluate and interpret performance
End

Figure 4.1: Fault detection in Scania Trucks: Design of experiment.

hyperparameter	values
max_depth	$2, 3, 4, 5, 6, 7$
n_estimators	$4, 8, 16, 32, 64, 128$
learning_rate	$0.05, 0.1, 0.15, 0.2$
booster	gbtree, dart
min_child_weight	$1, 4, 16, 64$
subsample	$0.6, 0.7, 0.8, 0.9, 1$
colsample_bytree	$0.6, 0.7, 0.8, 0.9, 1$

Table 4.1: Fault detection in Scania trucks: Set of tuned hyperparameters for the XGBoost

We train and evaluate a large amount of XGBoost models with different hyperparameters on a subset of the training data using various metrics and a cross-validation. We compare how the metrics rank the different models and we select a set of candidate models — the models that are ranked as best by at least one metric. Afterwards we compare the candidate models in terms of precision and recall over various thresholds on rest of the training set — a validation set. Since we have a clearly defined cost function, we select as the best model (out of the candidate models) the one that achieves minimal total cost and we set it the corresponding decision threshold. Once we select the final model we evaluate it using the testing set and discuss its real-world performance.

The individual steps of the experiment are described in details below and illustrated in Figure 4.1.

4.2.3.1 Data Splitting

The data set is provided already split into a training and testing set, as described above. For our experiment we need one more set — a validation set — which we will use for selecting the best decision threshold. Therefore we split the original training set into new training set and a validation set with a ratio 4:1 (i.e. the new training set thus contains 48000 records and the validation set 12000 records). We perform the split in a stratified way so that the ratio of positive and negative samples remains the same in the new training and the validation sets. When speaking about a training set we will from now on refer to this new training set.

4.2.3.2 Candidate Models Selection

We use the training set and a random search algorithm with cross-validation to train and evaluate multiple XGBoost models with different hyperparameters. Table 4.1 shows the set of hyperparameters we select from. For evaluation we use 10-fold cross-validation and we evaluate each model with four metrics —

AUROC, AUPRG, F1 score and accuracy — and we calculate a mean score of the respective metrics over the testing folds. F1 score and accuracy are both calculated on the predictions made by using the decision threshold equal to 0.5.

Training and evaluation of one model takes approximately one minute — the cross-validation (10 folds) for one set of hyperparameters thus takes approximately 10 minutes. As we have available 32 CPU cores we run 160 iterations of the random search (5 for each CPU) so that it takes approximately 50 minutes. As a result we obtain a list of 160 models each assigned four scores and four ranks (each rank in range $[1, 160]$ with rank 1 being the best).

To compare the rankings of the models by the evaluation metrics we visualize a pairplot of the models' ranks.

Finally, from the trained models we select a set of candidate models. A candidate model is a model that is ranked as the best model by at least one of the evaluation metrics.

4.2.3.3 Final Model and Decision Threshold Selection

We retrain every candidate model using the whole training set and we use the validation set to calculate and visualize precision, recall and total cost (calculated as \$ 10 for FP and \$ 500 for FN) over various decision thresholds. As the final model we select the model with the lowest total cost and we set it the corresponding decision threshold.

4.2.3.4 Evaluation and Performance Interpretation

We use the testing set to calculate accuracy, precision, recall and total cost. We interpret the metrics and discuss how the model performs in real-world.

4.2.4 Results

Figure 4.2 shows a pair plot of rankings of all the XGBoost models. We can see that F1 and accuracy agree on the ranking of the models as well as AUROC and AUPRG do. However, the two pairs disagree with each other, i.e. both AUROC and AUPRG disagree with both F1 and accuracy. This is not very surprising as the F1 and accuracy are based only on predictions at threshold 0.5 whereas AUROC and AUPRG evaluate the model over all the thresholds.

We identify two candidate models, i.e. models that are ranked by at least one metric as the best model Table 4.2 shows their ranks and evaluation scores and the XGBoost's hyperparameters. As expected, one of the models, model A, is selected by AUROC and AUPRG while the other, model B, is selected by F1 and accuracy. Based on the XGBoost's hyperparameters we can see that the model A has lower complexity than the model B. The model A has the same number of estimators (trees) but it has lower maximum depth and

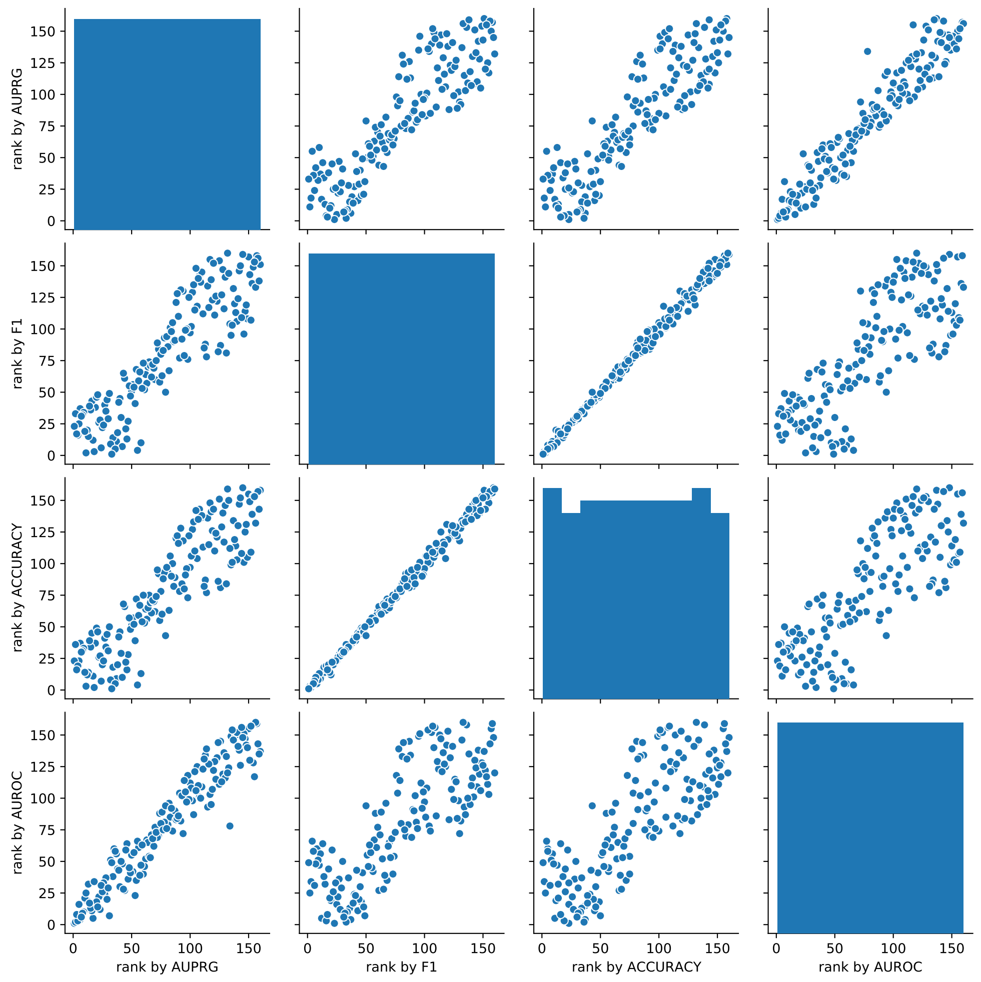
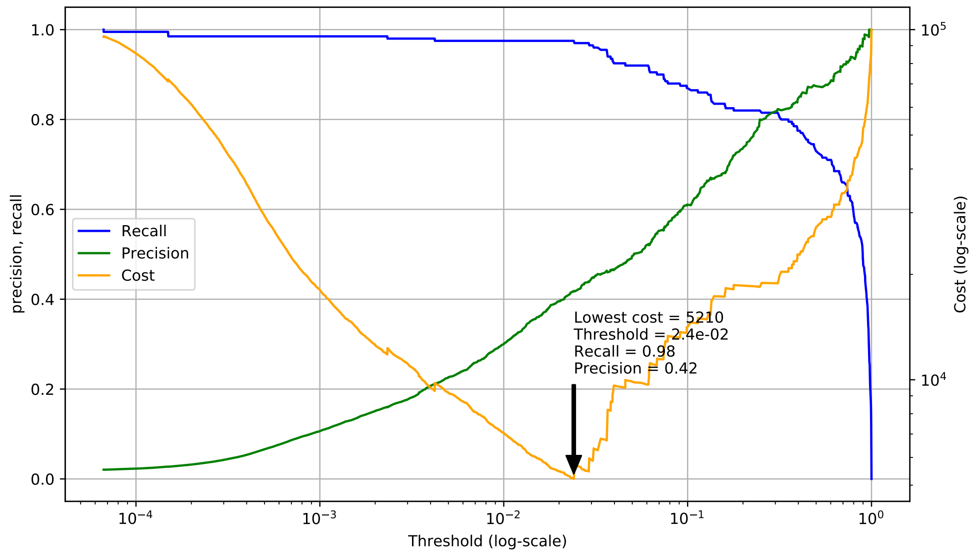


Figure 4.2: Fault detection in Scania Trucks: Pair plot of the evaluation metrics obtained from the random search.

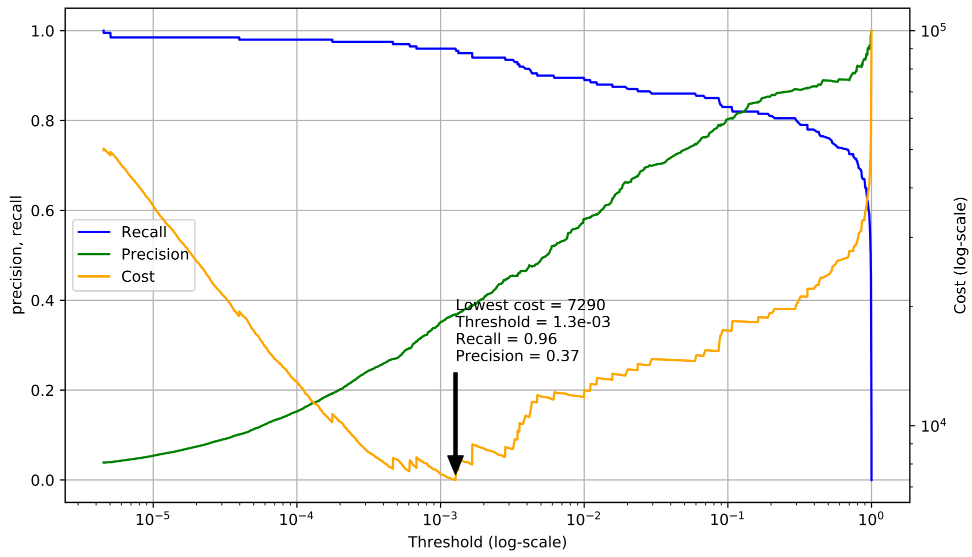
higher minimal child weight[12]. The lower complexity models are always more preferable as they have a lower risk of being overfitted. Therefore, from the current view we assume the model A as better so far.

Figure 4.3 shows plots of costs and precision and recall scores over various decision thresholds for both of the candidate models. We can see that the optimal threshold, i.e. the threshold where the cost is minimal, is very low and thus the recall is significantly higher than the precision. This is because the cost of the FNs is much higher than of the FPs and thus it is better to have as few FNs as possible, i.e. having a high recall. The lowest costs per model (annotated in the figure) are as follows:

[12]the minimal child weight defines the minimal number of samples in the node so that the node can be further split



(a) Model selected by AUROC and AUPRG



(b) Model selected by F1 and accuracy

Figure 4.3: Fault detection in Scania trucks: Precision-recall-cost plot for the candidate models.

	candidate model	
	A	B
rank by AUPRG	1	33
rank by F1	23	1
rank by ACCURACY	23	1
rank by AUROC	1	49
AUPRG score	0.999812	0.999767
F1 score	0.808412	0.831349
ACCURACY score	0.994146	0.994875
AUROC score	0.990641	0.988216
XGBoost param:subsample	0.9	0.8
XGBoost param:n_estimators	256	256
XGBoost param:min_child_weight	16	1
XGBoost param:max_depth	5	7
XGBoost param:learning_rate	0.1	0.2
XGBoost param:colsample_bytree	0.7	0.9
XGBoost param:booster	dart	dart

Table 4.2: Fault detection in Scania trucks: Ranks, scores and parameters of the candidate models

- model A: \$ 5210 at decision threshold $2.4e^{-2}$;
- model B: \$ 7290 at decision threshold $1.3e^{-3}$.

The results thus confirm that the model having the lower complexity, the model A, selected by AUPRG and AUROC, is better. Therefore, we select the model A as our final model and we set its decision threshold to $2.4e^{-2}$.

The evaluation of the final model (model A) on the testing set and using the decision threshold $2.4e^{-2}$ gives following results:

- Recall: 0.99
- Precision: 0.17
- Cost: 18950

The recall can be translated as that the model will detect 99 % of faults. The precision can be translated as that only 17 % of positive predictions will actually correspond to a faulty truck, or in other words 83 % of the predictions will be false alarms.

4.2.5 Discussion

In this experiment we demonstrated how to build a binary classification model for detection of point-based faults of Scania trucks. The results of the experiment show that AUPRG and AUROC metrics were better choice for model selection than F1 and accuracy metrics this data set and the XGBoost model. That is because there might be different importance of FP and FN (in our experiment we needed to have fewer amount of FN than FP) and thus it might be better to select a model that performs well at all thresholds and leave the decision threshold selection for later, when the specific domain needs are known.

Regarding evaluation of the model's performance in practice, we demonstrated that precision and recall can nicely interpret the model's performance, i.e. the probability that a fault will be detected and how often will the model predict a false alarm. It is good to note though, that the metrics cannot take into account any time information, e.g. how early will be the faults detected, as the point-based data do not contain any time information. To include the temporal information in evaluation, one has to use either data with range-based faults or another PdM approach.

4.3 Experiment — Failure Prediction in Azure Telemetry Data Set

This section describes an experiment where we demonstrate failure prediction approach and we compare its evaluation metrics. Failure prediction is an approach where the goal is to build a model that predicts whether a failure will happen in near future — in the monitoring window. This approach is suitable in cases when there are available data about failures and when the failures are expected to be preceded by a faulty behaviour of the subject.

The modeling typically consist of formulating the problem as a binary classification where before training the classifier, the samples prior to the failure are artificially labeled as positive. This, however, introduces challenge in the model's evaluation as there are more positive samples than failures. In Section 3.2.3 we described how classical precision and recall metrics can be modified so that they provide more realistic scores. We call the modified metric event-based precision and event-based recall. In this experiment, our goal is to compare the classical and event-based metrics in terms of model selection, the model's decision threshold selection and interpretability.

As said, failure prediction consists in predicting whether a failure will happen in the monitoring window. The size of the monitoring window can be either predefined by the domain (e.g. it might be known that the faulty behaviour of the subjects lasts no longer than 7 days before the failure) or it can be tuned as a hyperparameter. For the purpose of this experiment,

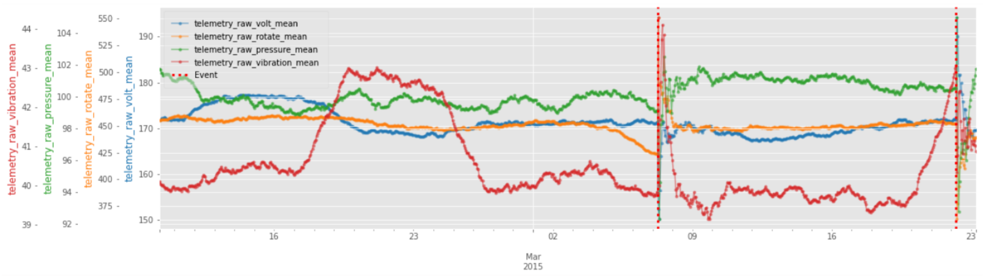


Figure 4.4: Failure prediction in Azure data set: Example of one machine's data. The vertical dotted lines represent the failure events.

we choose a data set that has the size of the monitoring window already predefined by the domain experts. The data set we chose is a publicly available Azure AI Gallery data set [45] which contains multiple data sources like sensor measurements and failure logs about 100 machines and the authors of the data set clearly define the task: predict whether a failure will happen in next 24 hours.

4.3.1 Data Set Description

The data set we use in this experiment is an Azure AI Gallery Predictive Maintenance data set [45] which contains continuously collected condition monitoring data and failure labels of an unspecified machinery. The data consist of telemetry data, error logs, maintenance logs and failure logs for 100 machines collected during whole year of 2015. The telemetry data include voltage, rotation, pressure and vibration measurements and are collected on an hourly basis — one value per hour. The error log contains time stamped information about non-breaking errors. The maintenance log contain time stamped events of both scheduled maintenance actions (regular inspection) and unscheduled maintenance actions (failures) . The failure logs contain time stamped information when the failures happened. When a failure happens on the machinery the failure is repaired and the machinery is put to operation again.

Preprocessing The data are available as separate CSV files for telemetry data and error, maintenance and failure logs. As the telemetry data are available on an hourly basis we round the time stamps of all the events, i.e. maintenance actions, errors and failures, to the closest hour and join the data on time stamps and machine identifications. To help the classifier identify temporal patterns in the data we create following time-based features (for every time point):

- mean, variance and sum of the telemetry data for the past 7 days;

- time from the last maintenance action, from the last error and from the last failure.

Figure 4.4 shows an example of the telemetry data for one machine.

4.3.2 Task Definition

The task, defined by the authors of the data set, is to predict whether a failure will happen in next 24 hours. The authors do not mention any warning window necessary for the predictions to be useful (e.g. so that there is enough time for the maintenance to be scheduled). Though it might be that there is no warning window necessary, we assume that is highly unlikely in practice and we assume the authors of the data set probably have not thought about the possibility of defining a warning window. Therefore, we set the warning window ourselves to 8 hours, i.e. one third of the monitoring window. This means that during the evaluation, all the predictions made less than or equal to 8 hours prior to the failure will be ignored. The size of our prediction window, i.e. the size of an interval prior to the failure where the training samples are considered as positive, is thus 16 (monitoring window minus warning window).

4.3.3 Design of Experiment

We approach the task as a supervised binary classification problem where we artificially label all the samples 24 hours (size of monitoring window) prior to each failure as positive to train the model. Regarding evaluation of the model, we are mainly interested in precision and recall, i.e. a probability that a true prediction actually predicts the failure and a probability of predicting a failure. However, as there are more true positive samples than the amount of failures it is not straightforward way how to use these metrics. In Section 3.9 we described a concept of event-based precision and recall where the TPs are replaced by detection scores and FP are replaced by discounted FP. Therefore, we design our experiment as to compare how these event-based metrics affect the model selection and decision threshold selection in comparison with the classical precision and recall.

The event-based metrics use detection score which has four parameters that can be set based on the domain specific needs. The parameters are a weight between existence and overlap (α), a cardinality function (γ), an overlap function (ω) and a positional bias function (δ). For more details about the parameters see Section 3.1.3. For our task, we set $\alpha = 0.8$ as we are rather interested in the existence of a true prediction in the prediction window than the amount of overlap. The cardinality, i.e. whether the predictions are fragmented, does not matter in failure prediction that much and thus we set it to be always equal to one. As an overlap function we use standard suggested definition in Section 3.1.3 and we set the positional bias to flat — i.e. we do not distinguish whether the prediction is at the beginning of the prediction window or at the end. As result, the detection score of every even is thus either equal to 0 (when there are no predicted positive samples) or is in range $[0.8125, 1]$[13], depending on the amount of positive predictions in the prediction window.

As the classification algorithm we choose gradient boosted trees, more specifically XGBoost [36], which is capable of predicting probabilities of classes and thus the decision threshold can be tuned.

Compared to classical binary classification, in failure prediction the predictions can be smoothed. It can for example happen that the model will predict a lone positive prediction among negative predictions which can be caused for example by a noise in the data. Therefore, the predictions are typically smoothed e.g. using a rolling mean where the predicted probability of each samples is calculated as the mean of several past predictions (including the current). For more details about prediction smoothing see Section 3.2. In this experiment, we use the rolling mean for smoothing of predicted probabilities and we tune the smoothing window size as a hyperparameter.

We design the experiment to consist of three steps (illustrated in Figure 4.5):

- data splitting — split the data into training and testing set;

- candidate models selection — use the training set and cross-validation to train and evaluate XGBoost model with different hyperparameters and smoothing window sizes, compare how the metrics rank the trained models and select a set of candidate models, i.e. models that are ranked at least by one metric as best;

- PR analysis — analyze candidate the models on the testing set using both classical and event-based precision and recall, discuss which model is the most suitable for the given task and compare the metrics' interpretability;

The individual steps of the experiment are in detail described below.

[13] 0.8 for the existence of a positive prediction in the prediction window plus $0.2 times 1/16$ (0.0125) for every positive prediction

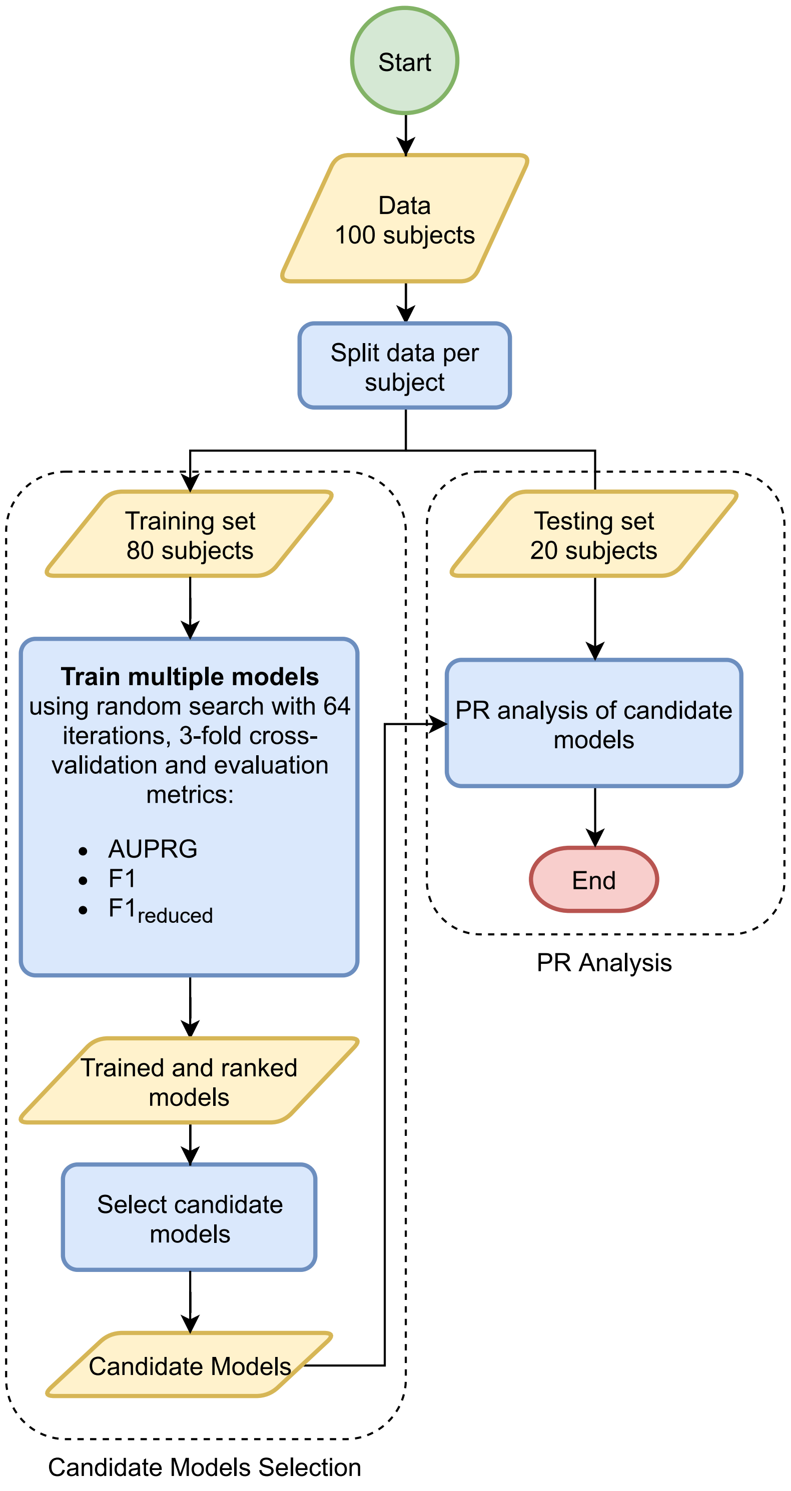


Figure 4.5: Failure prediction in Azure data set: Design of experiment

4.3.3.1 Data Splitting

We split the data into training and testing set. We identified two plausible splitting strategies: split by time and split by subject. The former consists in selecting the newest data (e.g. last two months, November and December) as the testing and the older data as the training. However, such splitting strategy might make the model to learn some subject specific patterns. Therefore, we adopt the latter splitting strategy: split by subject. We split the data set as follows:

- training set: 80 subjects
- testing set: 20 subjects

4.3.3.2 Candidate Models Selection

We use the training set to train and evaluate a large amount of XGBoost models with different XGBoost's hyperparameters and smoothing windows and we select a set of candidate models — models ranked as best by at least one metric.

As the evaluation metrics we use AUPRG, F1 score and event-based F1 score. The event-based F1 score is calculated based on event-based precision and recall metrics with the parameters as described above. The AUPRG (area under precision-recall-gain curve) is calculated based on classical precision and recall. We do not use the event-based metrics to calculate the area under event-based PR curve as the calculation of it is extremely computationally expensive. The regular AUPRG is calculated by sorting the samples by predicted score and the amount of true positive and false positive predictions can be then calculated using a cumulative sum operations [86]. Regarding the event-based metrics, however, the precision and recall have to be calculated separately for each threshold. The calculation of e.g. hundreds of thresholds then can take tens of minutes which is more than the amount of time for training the model itself. Therefore, we do not calculate the area under event-based precision-recall(-gain) curve and we use only the event-based F1 score calculated based on predictions made by the default decision threshold 0.5.

We run a random search algorithm with three-fold cross validation to train and evaluate the models. The average training time of one fold is 10 minutes. Since we have 32 CPU cores available we run 64 random search iterations so that the total computation time is approximately one hour. The models are then assigned an average score over the testing folds and are assigned a corresponding rank for every metric. Every trained model has thus assigned ranks per each metric in range $[1, 64]$ where rank 1 stands for the model with best score and rank 64 stands for the model with worst score.

hyperparameter	values
smoothing window	$\{1, 3, 5, 7\}$
XGBoost: max_depth	$\{2, 3, 4, 5, 6, 7\}$
XGBoost: n_estimators	$\{4, 8, 16, 32, 64, 128, 256\}$
XGBoost: learning_rate	$\{0.05, 0.1, 0.15, 0.2\}$
XGBoost: booster	{'gbtree', 'dart'}
XGBoost: min_child_weight	$\{1, 4, 16, 64\}$
XGBoost: subsample	$\{0.6, 0.7, 0.8, 0.9, 1\}$
XGBoost: colsample_bytree	$\{0.6, 0.7, 0.8, 0.9, \}$

Table 4.3: Failure prediction in Azure data set: Set of tuned parameters.

The hyperparameters we optimize are hyperparameters of the XGBoost algorithm and a size of the smoothing window. Table 4.3 summarizes all the tuned hyperparameters and the set of values we select from.

Once we have all the models evaluated we visualize a pairplot to compare ranks of the individual models for every pair of the four evaluation metrics. Afterwards, we select a set of candidate models from the best ranked models, i.e. models ranked high by at least one metric.

4.3.3.3 PR Analysis

We use testing set to perform PR analysis of the candidate models selected in previous step. For every candidate model we calculate both classical and event-based precision and recall over different thresholds and visualize them. We then discuss whether and how the candidate models differ and whether the classical and event-based metrics differ in decision threshold selection.

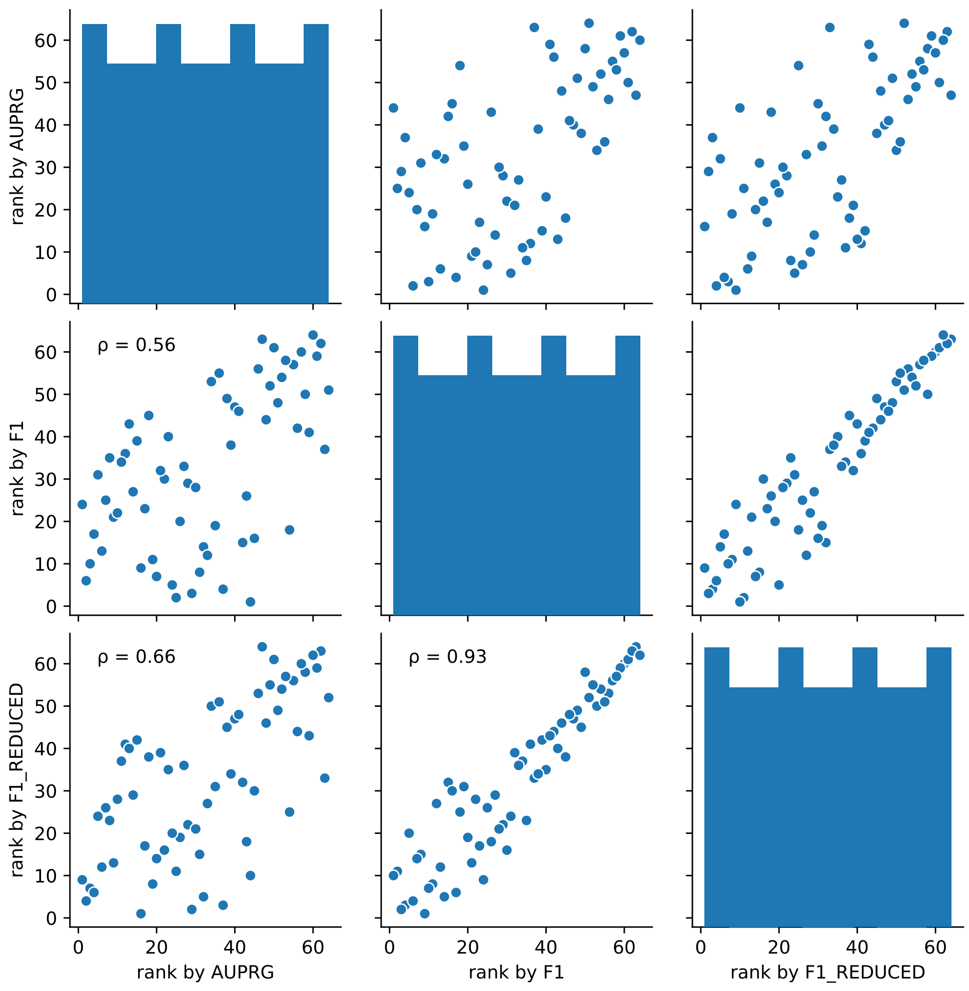


Figure 4.6: Failure prediction in Azure data set: Rankings of the models by various metrics based on the mean metrics' values on the testing cross-validation folds.

	candidate model		
	A	B	C
rank by AUPRG	1	16	44
rank by classical F1	24	9	1
rank by event-based F1	9	1	10
AUPRG score	0.999997	0.999991	0.999954
classical F1 score	0.95681	0.964553	0.969436
event-based F1 score	0.955075	0.961267	0.953946
param_smoothing_window	1	7	1
param_estimator__subsample	0.7	0.7	0.9
param_estimator__n_estimators	64	64	256
param_estimator__min_child_weight	16	16	1
param_estimator__max_depth	7	5	4
param_estimator__learning_rate	0.15	0.2	0.2
param_estimator__colsample_bytree	0.6	0.6	0.9
param_estimator__booster	dart	dart	dart

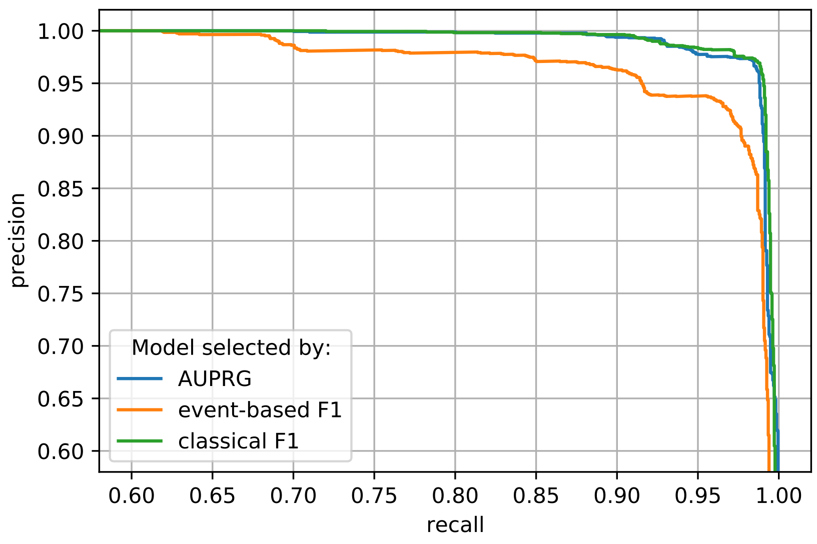
Table 4.4: Failure prediction in Azure data set: Ranks and parameters of the candidate models.

4.3.4 Results

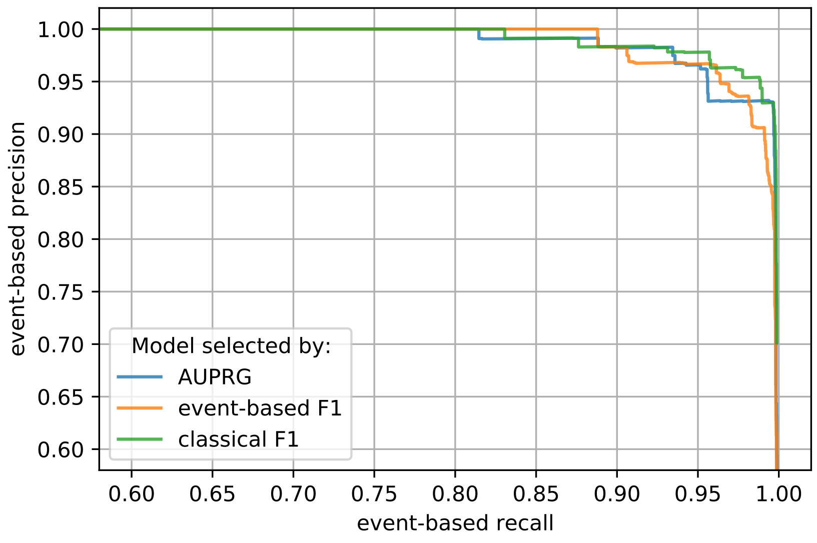
Figure 4.6 shows a pair plot of the rankings of the 64 trained models. We can see that F1 and event-based F1 relatively agree in the ranking though they select slightly different model as best. AUPRG, on the other hand, highly disagrees with both F1 and event-based F1. For example the best model selected by F1 has rank between 40 and 50 (with 64 being the worst) by AUPRG.

Table 4.4 shows scores, ranks and hyperparameters of the models ranked as best by at least one metric. We can see that the models chosen by F1 and AUPRG have both smoothing window of size 1 while the model chosen by event-based F1 has smoothing window of size 7. This might be caused by the low amount of lone FP, i.e. the smoothing only unnecessarily delays the positive predictions and thus causes the precision and recall scores to be. As both the classical and event-based F1 scores are very high — above 95 % — we can assume that most of the failures were predicted and that the predictions are made in the most of the prediction windows.

Figure 4.7 shows both the classical and event-based PR curves for all the three candidate models. We can see that the models selected by classical F1 score (green curve) and AUPRG (blue curve) are comparable in terms of classical PR curve. However, the model selected by classical F1 has significantly better event-based PR curve than the model selected by AUPRG. This is surprising as the model selected by F1 score was scored very low by the AUPRG



(a) Classical PR curves



(b) event-based PR curves

Figure 4.7: Failure prediction in Azure data set: Classical and event-based PR curves of the candidate models. Note, that both the x-axis and y-axis have range from 0.6 to 1.

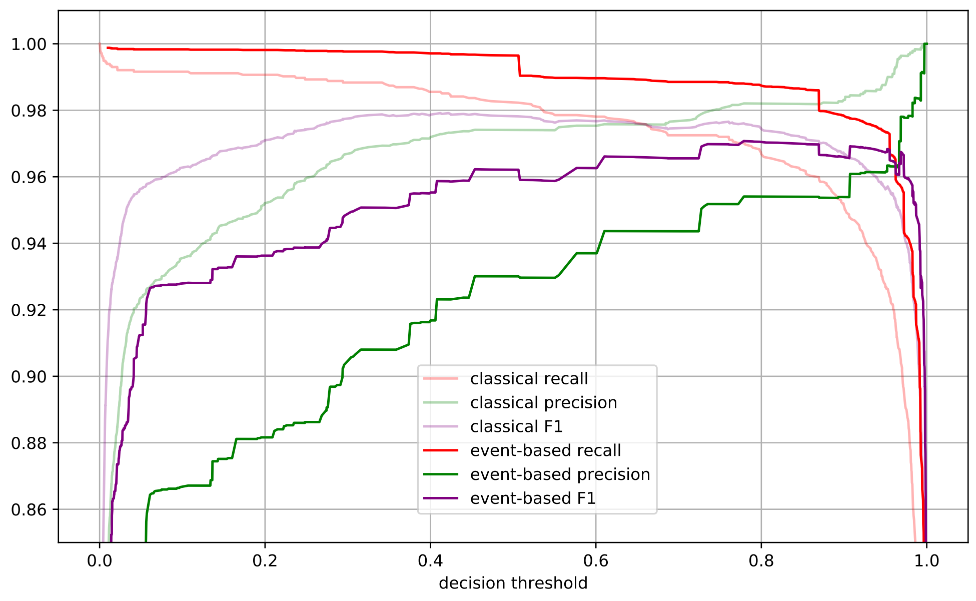


Figure 4.8: Failure prediction in Azure data set: Classical and event-based precision, recall and F1 scores over decision thresholds for the model selected by F1 score (model C). Note, that the y-axis has range from 0.86 to 1.

metric (rank 44 out of 64). It therefore suggests that a model that has good AUPRG score does not have to perform well regarding event-based metrics. Regarding the model selected by event-based F1 score, we can clearly see that it has significantly worse classical PR curve than the other two models. This is most probably caused by the smoothing — the predictions at the beginning of the prediction window might have low probabilities. Regarding event-based PR curve, however, we see that the model selected by event-based F1 score is slightly better at high event-based precision values than the model selected by classical F1 score. This suggests that the smoothing might help achieve better results when high precision is important, i.e. when the false alarms are costly.

If precision would be of high importance, we would choose the model selected by the event-based F1 score that smooths the predictions. However, since we do not know the exact costs of FP and FN, we choose the best performing model in overall. That is the model selected by classical F1 score as it has superior both classical PR and event-based PR curves over the other two models. Therefore, we use this model to compare how the classical and event-based differ in the decision threshold selection.

Figure 4.8 shows classical and event-based precision, recall and F1 scores over various decision thresholds for the model selected by classical F1 score. We can see that the event-based precision is significantly lower than the classical precision. That is caused by using the detection score instead of true positives and by using the discounted FP instead of standard FP. The size of the difference between the classical and event-based precision provides an insight into how are the FP close together — the higher the difference the more distant are the FP from each other. The event-based recall, on the other hand, is higher than the classical recall. That is caused by using the detection score with an existence reward, i.e. having only a single prediction in the prediction window causes the detection score to be $> \alpha$, i.e. at least as big as the existence weight. In our case $\alpha = 0.8$, as mentioned in the experiment design. This then leads to the highest value of event-based F1 score being at higher decision threshold (approx. 0.8) than the highest value of classical F1 score (approx. 0.4). In other words, the event-based metrics suggests that predicting only the samples that the model is more confident about as positive, i.e. selecting higher decision threshold, is likely to bring better precision without much of a decrease in recall.

4.3.5 Discussion

In this experiment we demonstrated failure prediction approach where we formulated the problem as a binary classification with an artificial labeling and we compared classical and event-based precision and recall metrics.

Regarding model selection, the results show that using event-based F1 score as a metric for model selection can select a model that has better recall at high precision values than models selected by classical F1 score or AUPRG. However, in other cases the model selected by classical F1 score was better. The results of model selection also suggest that when using event-based metrics it might be worth trying different sizes of artificial labeling, i.e. try to artificially label either less than or more than M[14] samples prior to the failure. In other words, the amount of artificial labeling might be another hyperparameter to tune.

Regarding decision threshold selection and interpretability, our results show that event-based precision and recall might provide more realistic estimates of the model's precision and recall. Moreover, using the event-based metrics for decision threshold selection might advise to select a higher decision threshold than when the classical metrics are used. This implies that according to event-based metrics, better precision can be achieved without much of a loss in recall. This can be especially useful information when the false alarms are costly and thus precision should be high.

Event-based metrics have several parameters that can be tuned such as the existence weight α or the positional bias function. It might be interesting to compare how the choice of these parameters affect the selection of the model. We consider this, however, as out of scope of this thesis.

[14]size of the monitoring window

4.4 Experiment — RUL Prediction of Turbofan Engines

This section describes an experiment where we demonstrate RUL prediction approach and we compare its evaluation metrics. RUL prediction is a PdM approach where the goal is to predict the exact time that remains until a failure occurs — the remaining useful life (RUL). This approach is suitable in cases when there is a continuous degradation process of the subject.

There exist two main approaches to RUL prediction: direct RUL prediction and HI-based RUL prediction. The first approach, the direct RUL prediction, consists in training a regression model to predict RUL values retrospectively calculated from run-to-failure data. The second approach, HI-based RUL prediction, consists in building a time series prediction model that predicts when a health indicator (HI) of the subject crosses a predefined failure threshold. The two approaches differ in what data are required for the model to be build (the necessity of run-to-failure data versus the necessity of a single HI and failure threshold) and in what models they use (classical regression vs time series prediction). In this experiment we choose to demonstrate the direct RUL prediction approach.

For the purpose of our experiment we choose a data set containing run-to-failure data of turbofan engines [46] which has already been used in many works regarding RUL prediction [70, 71, 87, 88] and is thus well known in the community of predictive maintenance and related fields. In contrast with the existing works, where the best performing model is selected based on one specific metric, most commonly RMSE, we evaluate the built models using multiple metrics and we analyze how do the models selected by various metrics differ.

4.4.1 Data Set Description

The data set used in this experiment is turbofan engine degradation data set that contains run-to-failure data of hundreds of turbofan engines of the same fleet. The data of each engine consist of multivariate time series containing measurements from 26 sensors and 3 operational settings. The time axis is the current number of an engine's operation cycle.

Each engine starts with different degree of initial wear and manufacturing variation which is unknown. This wear and variation is considered normal, i.e. it is not considered a fault condition. A fault develops at some point during the engine's operation and grows in magnitude until a failure occurs. The end of each time series thus marks the engine's failure. The average length of an engine's operation is approximately 200 cycles. Figure 4.9 shows an example of data of one engine.

Four different training data sets are provided with different operating conditions and different failure modes. Each training data set consists of the

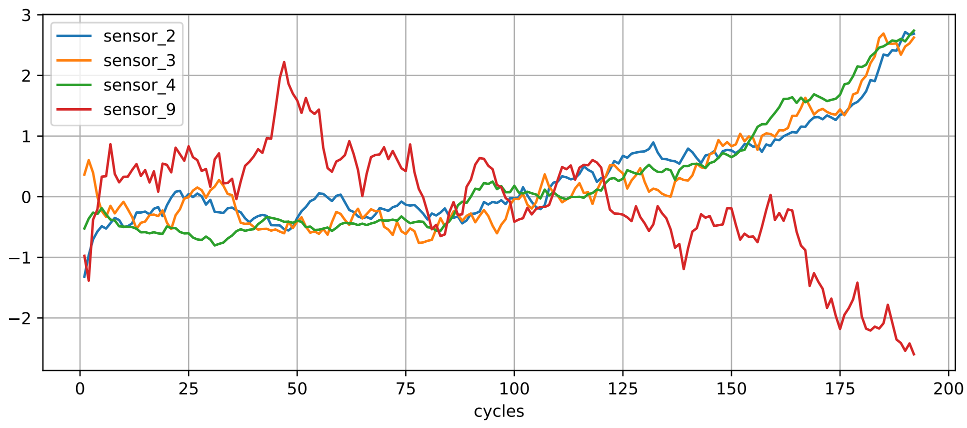


Figure 4.9: RUL prediction of turbofan engines: Sensor data for one engine. The failure of the engines occurred after the last operation cycle. We can see that a fault has developed somewhere between 50th and 100th cycle and it grows in magnitude until a failure.

multivariate time series (as described above) for hundreds of engines. There are four testing sets corresponding to the four training sets, respectively. Each testing set consists of two files. The first contains data from tens of engines in the same format as the training data but the data are randomly trimmed from the right side, i.e. the end of the time series does not mark the engines failure. The second file contains one RUL value for each of the engine in the first file corresponding to the length of the data trimmed from the right, i.e. the RUL at the end of each time series.

The data are provided as CSV files with the time series being in the long format, i.e. each row represents data for one operational cycle of one engine.

Target variable The data are run-to-failure with last record being the last measurement before the We failure occurred. We thus calculate the RUL retrospectively from the data such as for every subject the last record has RUL 1, second from the last record RUL 2 and so on.

Features As the features we use the sensor measurements, operational settings and current cycle number. Seven sensor features and one operational setting feature contain constant values. Therefore, we remove these features as they do not bring any information. The sensor measurements contain a lot of noise (see Figure 4.9 for illustration). Therefore we perform smoothing of the sensor values by a rolling mean of size 11 where each sensor value is replaced by the mean value of the past three values (including the current)[15].

[15]Note, that is is extremely important that the smoothing (or any other kind of rolling operation) must be done by assigning the aggregated value to the most-right element of the

At last, we normalize all the features using the values from the training data set.

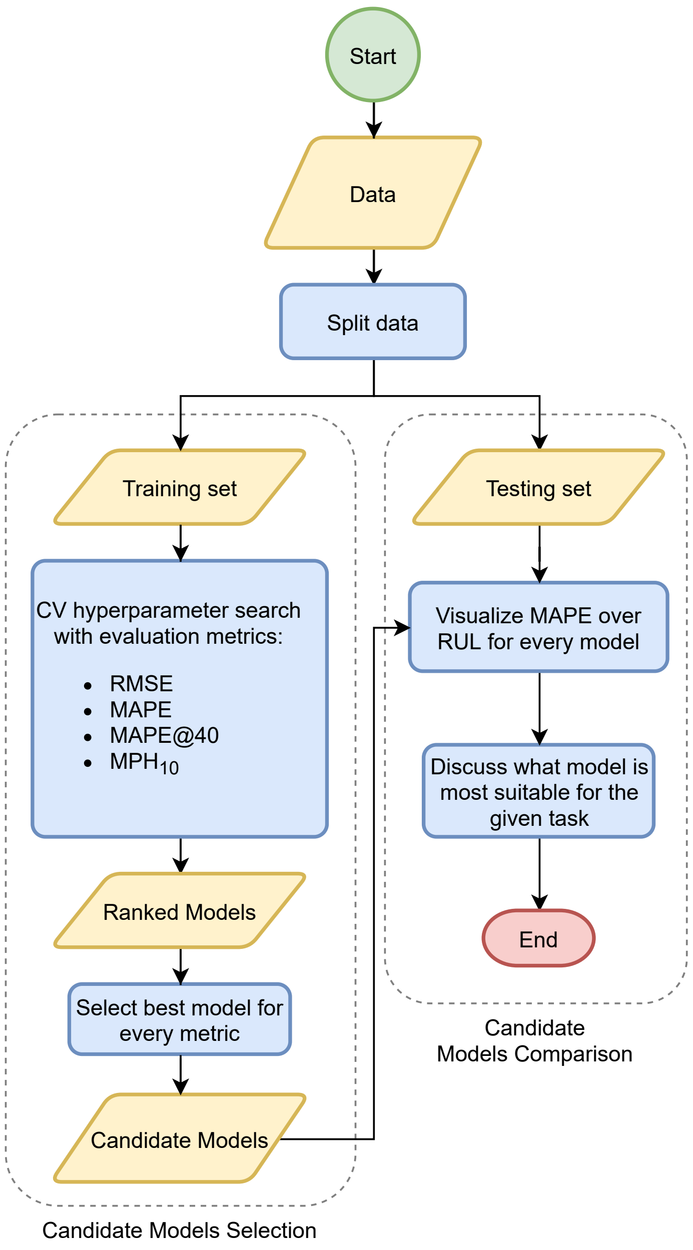

Start
Data
Split data
Training set
CV hyperparameter search with evaluation metrics:
• RMSE
• MAPE
• MAPE@40
• MPH10
Ranked Models
Select best model for every metric
Candidate Models
Candidate Models Selection
Testing set
Visualize MAPE over RUL for every model
Discuss what model is most suitable for the given task
End
Candidate Models Comparison

Figure 4.10: RUL prediction of turbofan engines: Design of experiment.

window. A centered window for example would use data from the future.

4.4.2 Task Definition

The task defined by the authors of the data set is to build a RUL prediction model using the training set and evaluate it using the testing set — i.e. to predict one RUL value for each engine in the testing set. However, we consider such evaluation as not being appropriate as a RUL prediction model deployed in production will most probably continuously predict RUL values at each operational cycle. We think a RUL prediction model should be rather evaluated by RUL predictions from the whole life cycles of the testing subjects rather than by only one randomly selected RUL. Therefore, we redefine the task as to:

- split the original training set (containing the condition monitoring data for each engine's whole life cycle) into new training and testing sets[16];
- use the newly created training set for building the model;
- use the newly created testing set for the model evaluation.

For the purpose of our experiment we use the first of the four provided training sets that contain data from exactly 100 engines that operate under stable conditions and develop a fault in the high pressure compressor.

Moreover, we set a warning window (W) so that the predictions close to the failures are excluded from the evaluation, i.e. all predictions made for actual RUL lesser than W are excluded from evaluation. There are two reasons for using the warning window. The first is that the predictions of RUL very near to 0 are likely to have high error (e.g. predicted RUL being 2 at actual RUL 1 is equal to error 200 %). The second reason is that predictions such close to the failure are commonly of low value in practice as the maintenance action can for example be already be scheduled when predicted RUL is for example 10 (rather than 1 or 2)[17]. As an average life cycle of the engines is about 200 cycles, we set the warning window to 5 cycles (2.25 % of the average life cycle).

4.4.3 Design of Experiment

The goals of this experiment are to demonstrate RUL prediction and compare its evaluation metrics. Commonly, there is a need for the RUL prediction models to have better performance with RUL reaching zero as the maintenance actions are typically performed when the subject's RUL is low (but not yet in the warning window). Therefore, we design the experiment so that the model selection is done in two steps. The first step consists in selecting a set of candidate models from a large set of trained models using various evaluation

[16]splitting should be then made per subject to avoid overfitting

[17]though this of course depends on the specific use case

metrics which provide a single score. The second step then consists in more an analysis of the candidate models in terms of how they perform relatively to RUL, i.e. whether for example some model predicts RUL accurately most of the time (thus having good overall performance) but has poor predictions near the failure. The individual steps of the experiment design (illustrated in Figure 4.10) are described below.

4.4.3.1 Data Splitting

Split the data into training and testing sets with ratio 4:1. Perform the split per subject, i.e. all data from one subject are either in the training or the testing set. We consider the split per subject is crucial so that the model does not learn some subject-specific patterns and so that we do not obtain overly optimistic evaluation results. The training and testing sets thus contain data about 80 and 20 subjects, respectively.

4.4.3.2 Candidate Models Selection

We use the training set to train the following regression models:

- XGBoost (regression version) with n_estimators $\in [3, 4, 5, 7, 8, 9]$ and max_depth $\in [16, 32, 64, 128, 256]$;
- SVR with gamma $\in [0.01, 0.1, 0.5, 1]$ and C $\in [0.1, 1, 10, 100]$;

and evaluate them using 4-fold cross validation[18]. We decide to evaluate each model using multiple metrics and later analyze whether they differ in what model they select as the best.

As described in the theoretical part of this thesis in Section 3.3.3, there exist many metrics and their variants how to evaluate a RUL prediction model. Therefore, we use only a small representative subset of the described metrics:

- root mean squared error (RMSE);
- mean absolute percentage error (MAPE);
- MAPE@40, i.e. MAPE calculated only on RUL values lower or equal to 40;
- MPH_{10}, i.e. mean prognostic horizon with α bound 10 — a mean time prior to a failure so that all the predictions differs from actual RUL by 10 at maximum.

We choose RMSE because it is one of the most commonly used metric in scientific articles [70, 71, 87, 88] and it gives the highest weight to the errors at

[18]split the data again per subject, i.e. each testing (training) fold contains 20 (60) subjects

high RUL values. Next, we choose MAPE as it should give more weight on the errors near the failure. At last, we choose metrics MAPE@40 and MPH_{10} which should select models only by their performance at low RUL values.

We try all the combinations of the above mentioned hyperparameters of the models totalling in 46 models (30 XGBoost models and 16 SVR models). We rank the models by mean score across the testing folds for each of the evaluation metrics so that trained model thus obtains 4 ranks all being $\in [1, 47]$. The lower the rank the better the model by the corresponding metric.

To compare how the evaluation metrics rank the models we visualize a pair plot of the rankings of the models. As the set of candidate models we then select the models that are ranked as the best by at least one of the metrics.

4.4.3.3 Candidate Models Comparison

We use the testing set to calculate MAPE relative to RUL for every candidate model, i.e. MAPE at RUL 50 is calculated from all the predictions made for samples with an actual RUL 50. We plot the errors, visually compare whether and how the candidate models differ and discuss which model may be more suitable for the given task and why.

4.4.4 Results

Figure 4.11 shows a pair plot of the ranks of the 46 trained models by the evaluation metrics. We can see that only pair of metrics that roughly agrees on the ranking is MAPE@40 and MPH_{10}, although MPH_{10} ranks several SVR models very high and MAPE@40 ranks them low. RMSE, MAPE and MPH agree on the best model, but highly disagree at the lower ranked models. MAPE@40, on the other hand, completely disagrees with RMSE and MAPE.

Table 4.5 shows parameters and evaluation results of the candidate models, i.e. models selected as best by at least one metric. We can see that there are two candidate models — the first, model A, selected by MAPE@40 and the second, model B, selected by the other three metrics. The first model has very low rank by MAPE and RMSE which is most probably caused by having very high errors at the RUL values higher than 40. From these single scores themselves, however, we can hardly interpret the models' performance as we do not know how it performs with respect to actual RUL values. Therefore, we perform the further analysis of the two models by plotting the errors relative to RUL.

Figure 4.12 shows MAPE metric relative to RUL values calculated using the testing set. Figure 4.13 then shows the candidate models' predictions for four random subjects. We can see that the SVR model (model B, selected by RMSE, MAPE and MPH) has consistently low MAPE at RUL values higher than 40 but the error then significantly rises with the RUL reaching 0. The XGBoost model (model A, selected by MAPE@40), on the other hand, has

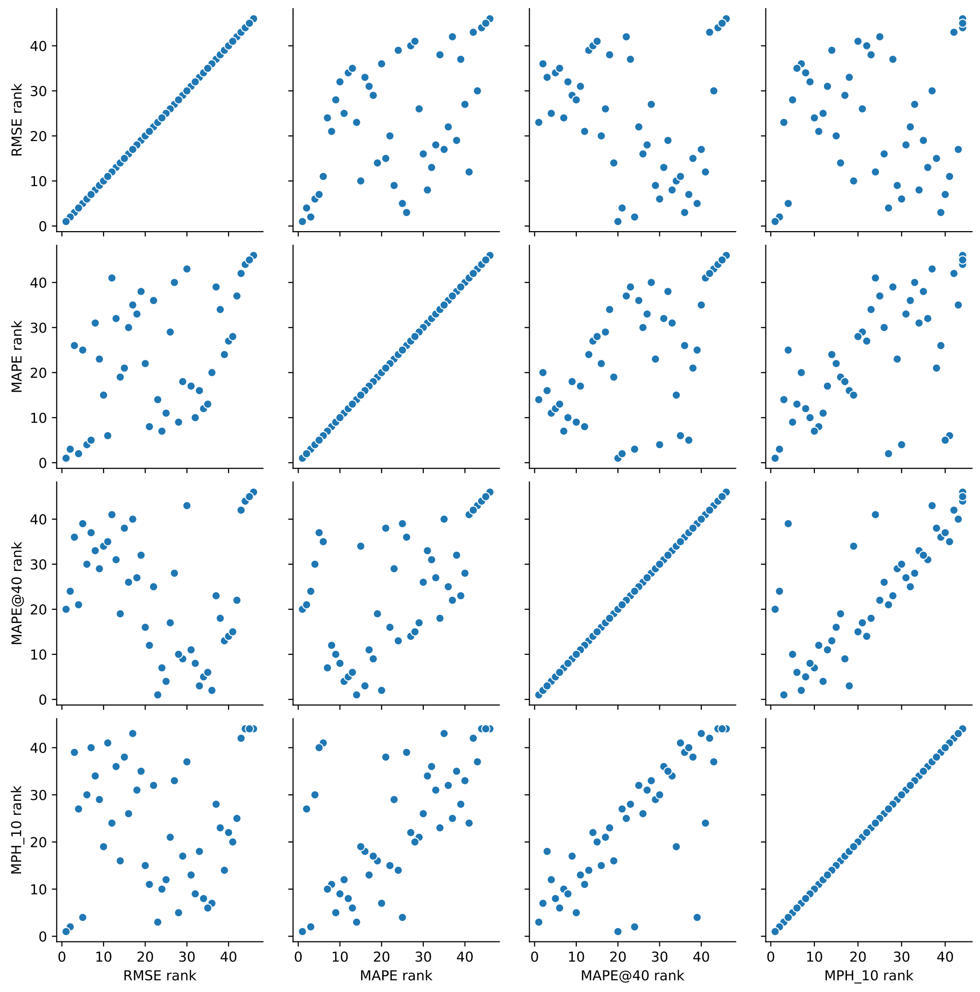


Figure 4.11: RUL prediction of turbofan engines: Pair plot of the ranks of the tested models.

higher errors at RULs higher than 40 but is much more accurate at lower values. However, we see that the standard deviation of the errors of the XGBoost model is significantly higher than of the SVR model. From the predictions on the four subjects, we can see that the predictions of XGBoost indeed are very unstable over various RUL values. Therefore, we consider the SVR model to be better, even though it has slightly worse predictions at low RUL values, as we consider the variance of the XGBoost model to be unsuitable for a practical application.

	candidate model	
	A	B
rank by RMSE	24	1
rank by MAPE	14	1
rank by MAPE@40	1	20
rank by MPH_10	3	1
RMSE	42.9	37.6
MAPE	30.3	27.5
MAPE@40	38.4	42.3
MPH_10	29.8	32.8
regressor	XGBoost	SVR
hyperparameters	n_estimators: 16 max_depth: 7	C: 100 gamma: 0.1

Table 4.5: RUL prediction of turbofan engines: the candidate models.

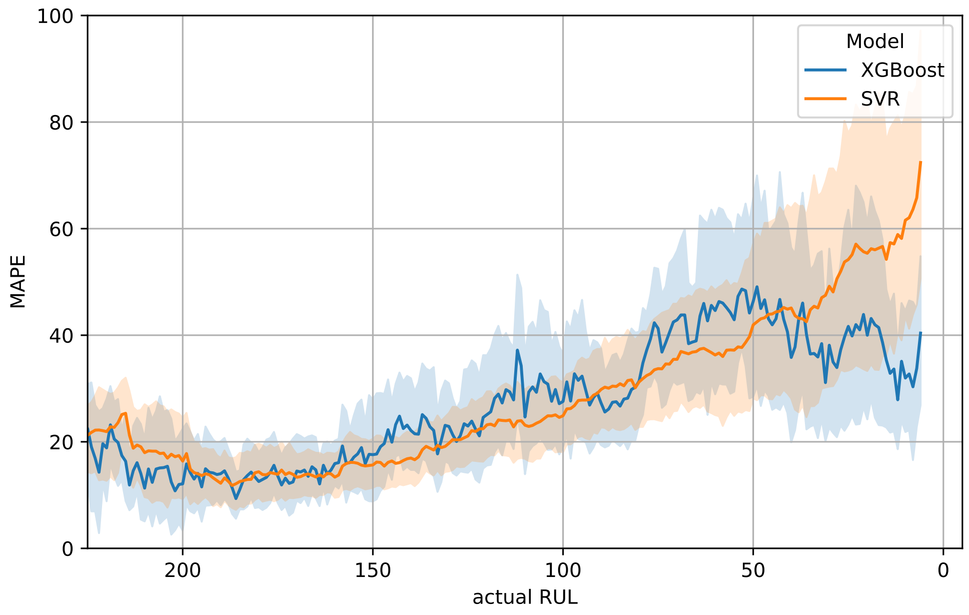


Figure 4.12: RUL prediction of turbofan engines: MAPE over various RUL values on the testing data set.

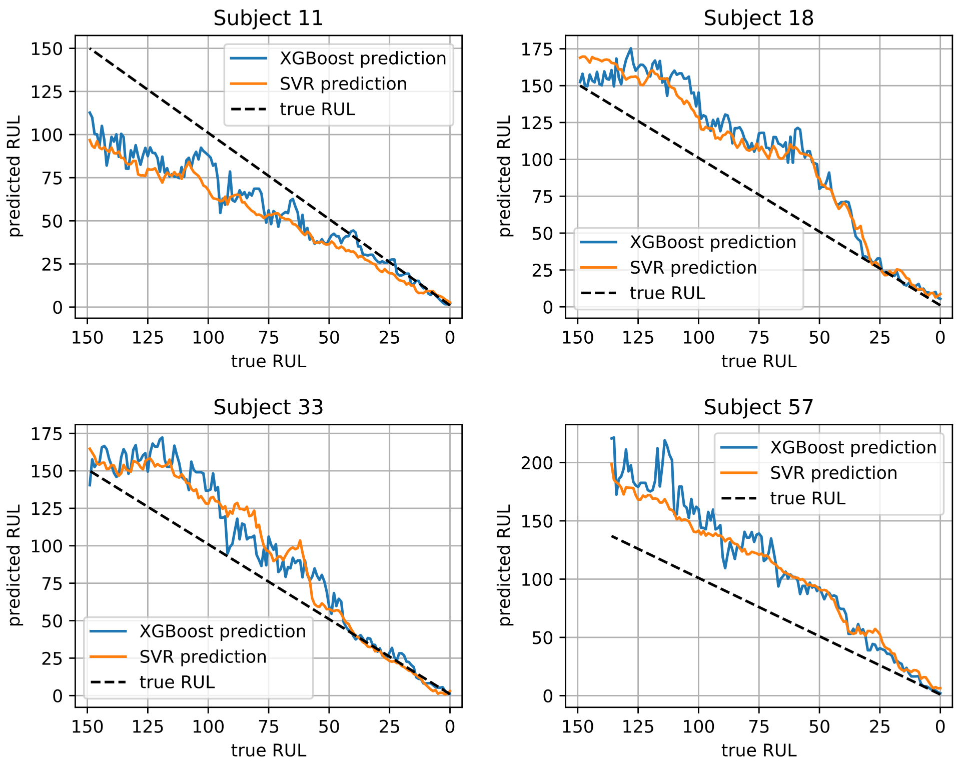


Figure 4.13: RUL prediction of turbofan engines: examples of predictions for multiple subjects

4.4.5 Discussion

In this experiment we demonstrated RUL prediction on a run-to-failure data set of turbofan engines using a direct RUL prediction, i.e. regression-based, approach. Our results show that RMSE and MAPE might highly disagree in model ranking. This suggests RMSE being unsuitable for RUL prediction model selection as the RUL prediction model is typically required to be more precise with RUL reaching lower values. Moreover, we demonstrated that using a single score to evaluate the whole model does not give necessary insight to evaluate its real-world performance. Rather, the errors should be plotted against actual RUL values as this might reveal that e.g. the model has very high variance of predictions over actual RUL values. That might for example reveal that the prediction has high variance over time which might make the model untrustable in practice — a model that at one cycle predicts RUL 80 cycles and the second cycle predicts RUL 40 cycles (XGBoost model predictions for subject 11 at actual RUL 65 in Figure 4.13) is very likely not to be trusted by the users.

Conclusion

PdM can utilize AI techniques to build a model that predicts condition of a subject. Depending on the available data and the degradation profile of the subject, various approaches how to predict the condition, i.e. approaches to PdM, can be used. In this thesis, we provided an overview of three different approaches — fault detection, failure prediction and remaining useful life prediction. In the theoretical part of this thesis, we theoretically described the approaches including their use case, data specifications, AI modeling techniques and evaluation metrics. In the practical part of the thesis, we conducted experiments on real-world publicly available data sets where we demonstrated all the approaches and compared their evaluation metrics.

The first approach, fault detection, is an approach suitable when there are no information about the actual failures of the subjects. It can be modelled as a binary classification or an anomaly detection. We identified two types of data for fault detection — data with point-based faults and data with range-based faults. Moreover, we described that for each of the types of data different evaluation metrics should be used, namely classical precision and recall and range-based precision and recall. In the experiments, we demonstrated detection of point-based faults in air-pressurized systems of Scania trucks. The authors of the data set defined a cost function for false alarms and missed faults where the cost of missing a fault was significantly higher. The results of the experiment showed that AUROC and AUPRG were more suitable for model selection than accuracy or F1 score calculated at a fixed decision threshold. Moreover, the results show that precision and recall can be nicely translated into a real-world performance.

The second approach, failure prediction, is an approach where the goal is to predict whether a failure will happen in near future — in a monitoring window. In the theoretical part, we described how it can be formulated as a binary classification problem and that it can be modeled by artificially labeling the samples in the training data prior to the failure as positive. Regarding evaluation, we explained why classical classification metrics might be unsuitable for evaluation of failure prediction. We described how two other types of precision and recall metrics used in literature, reduced and range-based, can be used to mitigate the issues of classical precision and recall. We proposed a new definitions of precision and recall, which we call event-based, as a combination of the two former mentioned. In the practical part, we conducted an experiment demonstrating failure prediction in Azure telemetry data set where the task was to predict whether a failure will happen in next 24 hours. We compared classical and event-based precision and recall metrics in terms of model selection, decision threshold selection and interpretability. The results show, that the classical metrics might be sufficient for model selection but the event-based metrics may provide more realistic interpretation of the model's performance and they advise to select higher decision threshold, i.e. higher recall can be achieved with low or none decrease in precision. The event-based metrics have several parameters that they inherit from range-based metrics such as weight of existence reward or a positional bias function. Moreover, the amount of artificial labeling can be taken as a hyperparameter in model selection. Therefore, we see a space for future research in comparing how the choice of the event-based metrics' parameters affects model choice and whether choosing the amount of artificial labeling higher or lower than the monitoring window size might result in better model's performance.

The last approach we cover, RUL prediction, is an approach where the goal is to predict the exact time left until the subject fails. In the theoretical part, we described two modeling approaches to RUL prediction, direct RUL prediction and HI-based RUL prediction, and we described various metrics how can be a RUL prediction model evaluated. In the practical part, we demonstrated direct RUL prediction approach on a turbofan engine degradation data set. We compared several metrics in terms of model selection and interpretability. The results show that RMSE and MAPE disagree in ranking of the models. This might be crucial for the practical application as it might be required for a RUL model to be more accurate as the time of a failure approaches. Moreover, we demonstrated that it might be essential for the practical evaluation of a RUL model to not only calculate one score but also to calculate the errors relative to RUL, e.g. visualize the errors relatively to how soon a failure will occur.

It would be interesting to extend the scope of our experiments by using more data sets and more ML models, preferably of different families. We assume that both the data sets and the ML models might have effect on the behaviour of the evaluation metrics. However, we consider this as beyond the scope of this thesis.

Bibliography

[1] Flach, P.; Kull, M. Precision-recall-gain curves: PR analysis done right. In *Advances in neural information processing systems*, 2015, pp. 838–846.

[2] Bilosova, A.; Bilos, J. Vibration diagnostics. *Technical University of Ostrava*, 2012.

[3] Lei, Y.; Li, N.; et al. Machinery health prognostics: A systematic review from data acquisition to RUL prediction. *Mechanical Systems and Signal Processing*, volume 104, 2018: pp. 799–834.

[4] JEFF SCHRIER. Wind turbine failure. 2013. Available from: `https://www.enr.com/articles/42352-are-four-wind-turbine-failures-in-five-weeks-too-many-for-nextera-energy`

[5] Ran, Y.; Zhou, X.; et al. A Survey of Predictive Maintenance: Systems, Purposes and Approaches. *arXiv preprint arXiv:1912.07383*, 2019.

[6] Sanger, D. Reactive, Preventive & Predictive Maintenance: IVC Technologies. Jan 2019. Available from: `https://ivctechnologies.com/2017/08/29/reactive-preventive-predictive-maintenance/`

[7] Lukány, J. *Application of Fourier and Wavelet Transform for Vibration and Acoustic Analysis of Machinery*. B.S. thesis, České vysoké učení technické v Praze. Vypočetní a informační centrum., 2018.

[8] Bechhoefer, E. High speed gear dataset. Available from: `http://data-acoustics.com/measurements/gear-faults/gear-1/`

[9] Abhinav Saxena, C. C. L., Kai Goebel; Chang, F.-K. CFRP Composites Data Set. Available from: `https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/`

[10] Agogino, A.; Goebel, K. Milling Data Set. 2007. Available from: `https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/`

[11] Wesley Hines, J.; Usynin, A. Current computational trends in equipment prognostics. *International Journal of Computational Intelligence Systems*, volume 1, no. 1, 2008: pp. 94–102.

[12] Tatbul, N.; Lee, T. J.; et al. Precision and recall for time series. In *Advances in Neural Information Processing Systems*, 2018, pp. 1920–1930.

[13] Jayasinghe, L.; Samarasinghe, T.; et al. Temporal convolutional memory networks for remaining useful life estimation of industrial machinery. *arXiv preprint arXiv:1810.05644*, 2018.

[14] Louw, C.; Heyns, P. Remaining Useful Life Prediction and Uncertainty Modelling with Bayesian Deep Learning. *Condition Monitoring and Diagnostic Engineering Management*, 2018: p. 267.

[15] Miao, Q.; Xie, L.; et al. Remaining useful life prediction of lithium-ion battery with unscented particle filter technique. *Microelectronics Reliability*, volume 53, no. 6, 2013: pp. 805–810.

[16] Saxena, A.; Goebel, K.; et al. Damage propagation modeling for aircraft engine run-to-failure simulation. In *2008 international conference on prognostics and health management*, IEEE, 2008, pp. 1–9.

[17] Yuan; Binhang; et al. WaveletAE: A Wavelet-enhanced Autoencoder for Wind Turbine Blade Icing Detection. Oct 2019. Available from: `https://arxiv.org/abs/1902.05625v2`

[18] Peng, D.; Liu, Z.; et al. A novel deeper one-dimensional CNN with residual learning for fault diagnosis of wheelset bearings in high-speed trains. *IEEE Access*, volume 7, 2018: pp. 10278–10293.

[19] Kauschke, S.; Fürnkranz, J.; et al. Predicting cargo train failures: A machine learning approach for a lightweight prototype. In *International Conference on Discovery Science*, Springer, 2016, pp. 151–166.

[20] Murray, J. F.; Hughes, G. F.; et al. Machine learning methods for predicting failures in hard drives: A multiple-instance application. *Journal of Machine Learning Research*, volume 6, no. May, 2005: pp. 783–816.

[21] Prytz, R. *Machine learning methods for vehicle predictive maintenance using off-board and on-board data.* Dissertation thesis, Halmstad University Press, 2014.

[22] Liu, J.; Li, Y.-F.; et al. A SVM framework for fault detection of the braking system in a high speed train. *Mechanical Systems and Signal Processing*, volume 87, 2017: pp. 401–409.

[23] Xiao, W. A probabilistic machine learning approach to detect industrial plant faults. *arXiv preprint arXiv:1603.05770*, 2016.

[24] Zhang, W.; Yang, D.; et al. Data-driven methods for predictive maintenance of industrial equipment: A survey. *IEEE Systems Journal*, volume 13, no. 3, 2019: pp. 2213–2227.

[25] Lee, J.; Wu, F.; et al. Prognostics and health management design for rotary machinery systems—Reviews, methodology and applications. *Mechanical systems and signal processing*, volume 42, no. 1-2, 2014: pp. 314–334.

[26] Jia, X.; Huang, B.; et al. A review of PHM Data Competitions from 2008 to 2017: Methodologies and Analytics. In *Proceedings of the Annual Conference of the Prognostics and Health Management Society*, 2018, pp. 1–10.

[27] Jahnke, P. Machine learning approaches for failure type detection and predictive maintenance. *Technische Universität Darmstadt*, volume 19, 2015.

[28] Korvesis, P. *Machine learning for predictive maintenance in aviation.* Dissertation thesis, 2017.

[29] Tsui, K. L.; Chen, N.; et al. Prognostics and health management: A review on data driven approaches. *Mathematical Problems in Engineering*, volume 2015, 2015.

[30] Saxena, A.; Celaya, J.; et al. Metrics for offline evaluation of prognostic performance. *International Journal of Prognostics and health management*, volume 1, no. 1, 2010: pp. 4–23.

[31] Kauschke, S.; Janssen, F.; et al. On the Challenges of Real World Data in Predictive Maintenance Scenarios: A Railway Application. In *LWA*, 2015, pp. 121–132.

[32] Weiss, G. M.; Hirsh, H. Learning to predict rare events in categorical time-series data. In *International Conference on Machine Learning*, 1998, pp. 83–90.

[33] Friedman, J.; Hastie, T.; et al. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[34] Goodfellow, I.; Bengio, Y.; et al. *Deep learning.* MIT press, 2016.

[35] Rokach, L.; Maimon, O. Z. *Data mining with decision trees: theory and applications*, volume 69. World scientific, 2008.

[36] Chen, T.; Guestrin, C. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, New York, NY, USA: ACM, 2016, ISBN 978-1-4503-4232-2, pp. 785–794, doi:10.1145/2939672.2939785. Available from: `http://doi.acm.org/10.1145/2939672.2939785`

[37] Cortes, C.; Vapnik, V. Support vector machine. *Machine learning*, volume 20, no. 3, 1995: pp. 273–297.

[38] Chen, J.; Tsai, C.-A.; et al. Decision threshold adjustment in class prediction. *SAR and QSAR in Environmental Research*, volume 17, no. 3, 2006: pp. 337–352.

[39] Hand, D. J.; Till, R. J. A simple generalisation of the area under the ROC curve for multiple class classification problems. *Machine learning*, volume 45, no. 2, 2001: pp. 171–186.

[40] Mobley, R. K. *An introduction to predictive maintenance.* Elsevier, 2002.

[41] Feather, F.; Siewiorek, D.; et al. Fault detection in an ethernet network using anomaly signature matching. *ACM SIGCOMM Computer Communication Review*, volume 23, no. 4, 1993: pp. 279–288.

[42] Choi, E.; Schuetz, A.; et al. Using recurrent neural network models for early detection of heart failure onset. *Journal of the American Medical Informatics Association*, volume 24, no. 2, 2017: pp. 361–370.

[43] Ng, K.; Steinhubl, S. R.; et al. Early detection of heart failure using electronic health records: practical implications for time before diagnosis, data diversity, data quantity, and data density. *Circulation: Cardiovascular Quality and Outcomes*, volume 9, no. 6, 2016: pp. 649–658.

[44] Wang, J.; Zhang, L.; et al. A new paradigm of cloud-based predictive maintenance for intelligent manufacturing. *Journal of Intelligent Manufacturing*, volume 28, no. 5, 2017: pp. 1125–1137.

[45] Uz, F. B. Azure AI Gallery Predictive Maintenance Data Set. 2016. Available from: `https://gallery.azure.ai/Collection/Predictive-Maintenance-Implementation-Guide-1`

[46] Saxena, A.; Goebel, K. Turbofan Engine Degradation Simulation Data Set. 2008. Available from: `https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/`

[47] Ma, S.; Jiang, M.; et al. Temperature effect and thermal impact in lithium-ion batteries: A review. *Progress in Natural Science: Materials International*, volume 28, no. 6, 2018: pp. 653–666.

[48] Case Western Reserve University Bearing Data Center. Available from: `https://csegroups.case.edu/bearingdatacenter/pages/welcome-case-western-reserve-university-bearing-data-center-website`

[49] Dua, D.; Graff, C. UCI Machine Learning Repository. 2017. Available from: `http://archive.ics.uci.edu/ml`

[50] Saha, B.; Goebel, K. Battery Data Set. 2007. Available from: `https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/`

[51] Klausen, A.; Van Khang, H.; et al. Novel Threshold Calculations for Remaining Useful Lifetime Estimation of Rolling Element Bearings. In *2018 XIII International Conference on Electrical Machines (ICEM)*, IEEE, 2018, pp. 1912–1918.

[52] ISO. Mechanical vibration—Evaluation of machine vibration by measurements on non-rotating parts. Standard, International Organization for Standardization, Geneva, CH, 2000.

[53] Chen, D.-G.; Lio, Y.; et al. *Statistical modeling for degradation data*. Springer, 2017.

[54] Wei, M.; Chen, M.; et al. Multi-sensor information based remaining useful life prediction with anticipated performance. *IEEE transactions on Reliability*, volume 62, no. 1, 2013: pp. 183–198.

[55] Tony Lindgren, J. B. APS Failure at Scania Trucks Data Set. 2016. Available from: `https://archive.ics.uci.edu/ml/datasets/APS+Failure+at+Scania+Trucks`

[56] Widodo, A.; Yang, B.-S. Support vector machine in machine condition monitoring and fault diagnosis. *Mechanical systems and signal processing*, volume 21, no. 6, 2007: pp. 2560–2574.

[57] PHM Data Challenge 2015 — Power Plant Faults. 2015. Available from: `https://www.phmsociety.org/events/conference/phm/15/data-challenge`

[58] Nectoux, P.; Gouriveau, R.; et al. PRONOSTIA: An experimental platform for bearings accelerated degradation tests. 2012.

[59] Helwig, N. Condition monitoring of hydraulic systems Data Set. 2018. Available from: `http://archive.ics.uci.edu/ml/datasets/Condition+monitoring+of+hydraulic+systems#`

[60] Santos, P.; Villa, L. F.; et al. An SVM-based solution for fault detection in wind turbines. *Sensors*, volume 15, no. 3, 2015: pp. 5627–5648.

[61] Mahadevan, S.; Shah, S. L. Fault detection and diagnosis in process data using one-class support vector machines. *Journal of process control*, volume 19, no. 10, 2009: pp. 1627–1639.

[62] Zhao, Y.; Yang, L.; et al. Decision tree-based fault detection and classification in solar photovoltaic arrays. In *2012 Twenty-Seventh Annual IEEE Applied Power Electronics Conference and Exposition (APEC)*, IEEE, 2012, pp. 93–99.

[63] Guo, M.-F.; Zeng, X.-D.; et al. Deep-learning-based earth fault detection using continuous wavelet transform and convolutional neural network in resonant grounding distribution systems. *IEEE Sensors Journal*, volume 18, no. 3, 2017: pp. 1291–1300.

[64] Jia, F.; Lei, Y.; et al. Deep neural networks: A promising tool for fault characteristic mining and intelligent diagnosis of rotating machinery with massive data. *Mechanical Systems and Signal Processing*, volume 72, 2016: pp. 303–315.

[65] Janssens, O.; Slavkovikj, V.; et al. Convolutional neural network based fault detection for rotating machinery. *Journal of Sound and Vibration*, volume 377, 2016: pp. 331–345.

[66] Borovicka, T.; Jirina Jr, M.; et al. Selecting representative data sets. *Advances in data mining knowledge discovery and applications*, 2012: pp. 43–70.

[67] Chandola, V.; Banerjee, A.; et al. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, volume 41, no. 3, 2009: pp. 1–58.

[68] Ranjan, C.; Reddy, M.; et al. Dataset: rare event classification in multivariate time series. *arXiv preprint arXiv:1809.10717*, 2018.

[69] Mahamad, A. K.; Saon, S.; et al. Predicting remaining useful life of rotating machinery based artificial neural network. *Computers & Mathematics with Applications*, volume 60, no. 4, 2010: pp. 1078–1087.

[70] Babu, G. S.; Zhao, P.; et al. Deep convolutional neural network based regression approach for estimation of remaining useful life. In *International conference on database systems for advanced applications*, Springer, 2016, pp. 214–228.

[71] Peng, W.; Ye, Z.-S.; et al. Bayesian Deep-Learning-Based Health Prognostics Toward Prognostics Uncertainty. *IEEE Transactions on Industrial Electronics*, volume 67, no. 3, 2019: pp. 2283–2293.

[72] He, W.; Williard, N.; et al. Prognostics of lithium-ion batteries based on Dempster–Shafer theory and the Bayesian Monte Carlo method. *Journal of Power Sources*, volume 196, no. 23, 2011: pp. 10314–10321.

[73] Yoo, Y.; Baek, J.-G. A novel image feature for the remaining useful lifetime prediction of bearings based on continuous wavelet transform and convolutional neural network. *Applied Sciences*, volume 8, no. 7, 2018: p. 1102.

[74] Zhang, Y.; Xiong, R.; et al. Long short-term memory recurrent neural network for remaining useful life prediction of lithium-ion batteries. *IEEE Transactions on Vehicular Technology*, volume 67, no. 7, 2018: pp. 5695–5705.

[75] Nectoux, P.; Gouriveau, R.; et al. PRONOSTIA: An experimental platform for bearings accelerated degradation tests. 2012.

[76] Kluyver, T.; Ragan-Kelley, B.; et al. Jupyter Notebooks – a publishing format for reproducible computational workflows. 01 2016.

[77] Pedregosa, F.; Varoquaux, G.; et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, volume 12, 2011: pp. 2825–2830.

[78] Travis E, O. *A guide to NumPy*. USA: Tregol Publishing, 2006.

[79] Virtanen, P.; Gommers, R.; et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, volume 17, 2020: pp. 261–272, doi:https://doi.org/10.1038/s41592-019-0686-2.

[80] development team, T. M. Matplotlib: Python plotting — Matplotlib 2.2.2. Available from: `https://matplotlib.org`

[81] Waskom, M.; Botvinnik, O.; et al. mwaskom/seaborn: v0.10.1 (April 2020). Apr 2020, doi:10.5281/zenodo.3767070.

[82] McKinney, W. pandas: a foundational Python library for data analysis and statistics. *Python for High Performance and Scientific Computing*, volume 14, 2011.

[83] Python implementation of paper ”Precision and Recall for Time Series”. Available from: `https://github.com/KurochkinAlexey/Time-series-precision-recall`

[84] Meelis Kull, M. P. N., Telmo de Menezes e Silva Filho. pyprg: Python package for creating Precision-Recall-Gain curves and calculating area under the curve. Available from: `https://github.com/meeliskull/prg/tree/master/Python_package`

[85] Poetry. Available from: `https://python-poetry.org/`

[86] Kurbiel, T. How to efficiently implement Area Under Precision-Recall Curve (PR-AUC). 2020. Available from: `https://towardsdatascience.com/how-to-efficiently-implement-area-under-precision-recall-curve-pr-auc-a85872fd7f14`

[87] Mosallam, A.; Medjaher, K.; et al. Data-driven prognostic method based on Bayesian approaches for direct remaining useful life prediction. *Journal of Intelligent Manufacturing*, volume 27, no. 5, 2016: pp. 1037–1048.

[88] Ellefsen, A. L.; Bjørlykhaug, E.; et al. Remaining useful life predictions for turbofan engine degradation using semi-supervised deep architecture. *Reliability Engineering & System Safety*, volume 183, 2019: pp. 240–251.

Appendix A

Acronyms

AI	artificial intelligence.
ANN	artificial neural network.
AUPRG	area under precision-recall-gain curve.
AUROC	area under receiver operating characteristic curve.
CV	cross-validation.
EoUP	end of useful predictions.
FN	false negative.
FP	false positive.
FPR	false positive rate.
HI	health indicator.
LSTM	long short-term memory.
MAE	mean absolute error.
MAPE	mean absolute percentage error.
ML	machine learning.
MPH	mean prognostic horizon.
PDF	probability density function.
PdM	predictive maintenance.

PR	precision-recall.
PRG	precision-recall-gain.
RMSE	root mean squared error.
ROC	receiver operating characteristic.
RUL	remaining useful life.
SVM	support-vector machine.
SVR	support-vector regression.
TN	true negative.
TP	true positive.
TPR	true positive rate.

Appendix B

Contents of CD

```
│ text/ ... a directory with the thesis in PDF format and LaTeXsource code
└ experiments/ ........................... a directory with experiments
```