

Anomaly detection for the injection kicker magnets of the CERN Large Hadron Collider

Thiebout Dewitte

Thesis voorgedragen tot het behalen
van de graad van Master of Science
in de ingenieurswetenschappen:
computerwetenschappen, hoofdoptie
Computationele informatica

Promotoren:

Prof. dr. ir. H. Blockeel
Dr. ir. W. Meert

Assessoren:

E. Van Wolputte
Prof. dr. D. Hughes

Begeleiders:

E. Van Wolputte
Ing. P. Van Trappen

© Copyright KU Leuven

Without written permission of the thesis supervisors and the author it is forbidden to reproduce or adapt in any form or by any means any part of this publication. Requests for obtaining the right to reproduce or utilize parts of this publication should be addressed to the Departement Computerwetenschappen, Celestijnenlaan 200A bus 2402, B-3001 Heverlee, +32-16-327700 or by email info@cs.kuleuven.be.

A written permission of the thesis supervisors is also required to use the methods, products, schematics and programmes described in this work for industrial or commercial use, and for submitting this publication in scientific contests.

Zonder voorafgaande schriftelijke toestemming van zowel de promotoren als de auteur is overnemen, kopiëren, gebruiken of realiseren van deze uitgave of gedeelten ervan verboden. Voor aanvragen tot of informatie i.v.m. het overnemen en/of gebruik en/of realisatie van gedeelten uit deze publicatie, wend u tot het Departement Computerwetenschappen, Celestijnenlaan 200A bus 2402, B-3001 Heverlee, +32-16-327700 of via e-mail info@cs.kuleuven.be.

Voorafgaande schriftelijke toestemming van de promotoren is eveneens vereist voor het aanwenden van de in deze masterproef beschreven (originele) methoden, producten, schakelingen en programma's voor industrieel of commercieel nut en voor de inzending van deze publicatie ter deelname aan wetenschappelijke prijzen of wedstrijden.

Preface

In front of you lies the final work of an important chapter in my life: studying computer science. I have started this journey in 2014 as a student in informatics at KU Leuven Kulak and continued as a civil engineer in Leuven. In between I had an amazing exchange year at the Technical University of Munich.

I feel honoured to have worked on a project of this scale, together with CERN. I would like to thank the people without whom this would not have been possible. First, thank you to my daily advisors Elia (KU Leuven) and Pieter (CERN). During weekly meetings, they provided me with constructive feedback. Elia gave valuable insights and ideas about machine learning, whereas Pieter his domain knowledge about CERN was useful. Second, thank you to my promoter Prof. Hendrik Blockeel. During meetings he asked critical questions, which have ultimately led to an in-depth evaluation of this work. Also, thank you to my co-promoter Wannes Meert. Last but not least, I want to thank my family and friends. Thank you to my parents for supporting me in my studies, and to my sisters (especially Soetkin, for living together in Leuven). Thank you Matthieu for proof-reading this work and providing me with helpful feedback.

I hope you enjoy the read.

Thiebout Dewitte

Contents

Preface	i
Abstract	iv
Samenvatting	v
List of Figures and Tables	vi
List of Abbreviations and Symbols	ix
1 Introduction	1
2 Background	3
2.1 The Large Hadron Collider	3
2.2 Anomaly detection	6
2.3 Feature selection	10
2.4 COBRAS: interactive clustering with super-instances	10
2.5 Previous work	12
3 Data	13
3.1 General information	13
3.2 Types of data	13
3.3 Logbook	16
3.4 Summary	18
4 Problem statement	19
4.1 Problem	19
4.2 Goal	19
5 The anomaly detection application	21
5.1 Preprocessing	21
5.2 Anomaly detection	28
5.3 Postprocessing	29
5.4 Evaluation	30
5.5 Visualisation	30
6 Experiments and results	33
6.1 Global decisions	34
6.2 Grid search	34
6.3 Feature selection	40
6.4 Live evaluation	48

6.5	Incorporating expert feedback	52
7	Conclusion	61
7.1	Main contributions	61
7.2	Future work	62
A	List of all data collections	63
B	Feature selection: feature sets	65
B.1	GMM	65
B.2	iFOREST	66
C	Poster	69
	Bibliography	71

Abstract

Reliability, availability and maintainability are parameters that determine if a large-scale accelerator system can be operated in a sustainable and cost effective manner. Beam transfer equipment such as kicker systems are critical components with potential significant impact on the global performance of the entire machine complex. Identifying root causes of malfunctions is currently tedious, and will become infeasible in future systems due to increasing complexity. Machine learning can help to automate this process. For this purpose a collaboration between CERN and KU Leuven was founded.

Several iterations of the study have yielded an *anomaly detection pipeline* which includes preprocessing, detection, postprocessing and evaluation. Merging data of different, asynchronous sources is one of the main challenges. Currently, Gaussian Mixture Models and Isolation Forests are used as unsupervised detectors, but any detector can easily be plugged into the system. During evaluation, we compare the detector predictions to manual e-logbook entries, which constitute a noisy ground truth. Lastly, we try to incorporate expert knowledge by means of semi-supervised clustering with COBRAS. A grid search allows for hyper-parameter optimization across the entire pipeline, yielding promising results.

Samenvatting

Betrouwbaarheid, beschikbaarheid en onderhoudsvriendelijkheid zijn parameters die bepalen of een deeltjesversneller op een duurzame en kostenefficiënte manier gebruikt kan worden. Apparatuur om deeltjes te injecteren zoals een injectie kicker magneet is een essentiële component met mogelijks significante impact op de globale prestatie van het gehele machinecomplex. Het identificeren van de oorzaken van storingen is een langdradige taak, en zal in toekomstige systemen niet meer mogelijk zijn door de stijgende complexiteit van de machines. Machine learning kan helpen om dit proces te automatiseren. Met dit doel werd een samenwerking tussen CERN en KU Leuven opgericht.

Meerdere iteraties van deze studie hebben geleid tot een *anomalie detectie pipeline* die volgende stappen omvat: preprocessing, detectie, postprocessing en evaluatie. Een van de grote uitdagingen is het combineren van data uit verschillende, asynchrone bronnen. Momenteel worden Gaussian Mixture Modellen en Isolation Forests gebruikt als detectors, maar eender welke detector kan eenvoudig ingeplugd worden. Bij de evaluatie vergelijken we voorspellingen op basis van de detector met een manueel samengesteld e-logbook. Deze noisy labels vormen een tweede grote uitdaging. Ten slotte proberen we kennis van experts in rekening te brengen door semi-gesuperviseerde clustering met COBRAS. Een grid search laat toe om hyperparameters te optimaliseren over de gehele pipeline, met veelbelovende resultaten als gevolg.

List of Figures and Tables

List of Figures

2.1	The LHC tunnel (source: [15])	3
2.2	The CERN accelerator complex (source: [14])	4
2.3	Isolating x_i and x_o (source: [11])	8
2.4	Constraint-based clustering with COBRAS (source: [19])	11
5.1	The anomaly detection pipeline	21
5.2	Spectrogram: a frequency-intensity representation of a time signal . . .	23
5.3	Extracted frequency and intensity information	25
5.4	False negative due to a wrong filter	26
5.5	The effect of filtering	27
5.6	Segments based on STATE:MODE data	29
5.7	Visualisation of the predicted segments. From June 7 until June 14 (2017), there are two TPs and one FN. The other segments are TNs as is the majority of the segments. The anomalies are shown together with the PRESSURE and TEMP_MAGNET_UP data. There are spikes in the PRESSURE data close to the first two anomalies.	31
6.1	PR curves and predictions of the best results after a grid search with anomaly_detector=GMM, beam=1, segment_score=max, threshold=99%.	38
6.2	Grid search results for 2017 and 2018, with settings anomaly_detector=GMM, beam=1, segment_score=max.	39
6.3	PR curves and predictions of the best results after a grid search with anomaly_detector=IFOREST, beam=1, segment_score=max, threshold=95%.	41
6.4	Grid search results for 2017 and 2018, with settings anomaly_detector=IFOREST, beam=1, segment_score=max.	42
6.5	PR curves and predictions of the best results after feature selection with anomaly_detector=GMM, beam=1, segment_score=max, threshold=99%.	45
6.6	The progress of feature selection for both anomaly detectors on the data of beam 1.	46

6.7	PR curves and predictions of the best results after feature selection with <code>anomaly_detector=iFOREST</code> , <code>beam=1</code> , <code>segment_score=max</code> , <code>threshold=95%</code>	47
6.8	Live evaluation: set-up. For predictions during week t a model trained on historical data from week 0 until the last day of week $t - 1$ is used. At the beginning of week $t + 1$ the model is retrained incorporating the additional data of week t	48
6.9	The system queries the user: “ <i>Should segment X and segment Y be in the same cluster?</i> ”. Automatically generated links guide the expert to the segments in the explorer, where pressure and temperature datasets are pre-selected.	53
6.10	The FP does not show any spikes in the data. The data pattern is similar to the given TN. The expert should answer the query with <i>yes</i> , with the aim to cluster the FP as a TN.	54
6.11	Special case 1: both the FP and TP contain abnormal patterns in the pressure. The expert should answer the query with <i>yes</i> , with the aim to cluster the FP as a TP.	56
6.12	Special case 2: the FP contains a clear spike in the pressure, whereas the TN shows normal behaviour. The expert should answer the query with <i>no</i>	57

List of Tables

2.1	MKI magnet installation names	5
2.2	Confusion matrix	8
2.3	Computing rank	9
3.1	The time span of used data	13
3.2	Continuous data	14
3.3	IPOC data	15
3.4	State data	15
3.5	Values of state data	16
3.6	LHC data	16
3.7	Controller data	17
3.8	A logbook entry	17
3.9	The amount of labels	18
5.1	The specification of filters removing erroneous sensor measurements.	24
6.1	Optimal parameters for GMM per year as result of a grid search with <code>beam=1</code> , <code>segment_score=max</code>	37
6.2	A general set of parameters for GMM.	37
6.3	Optimal parameters for iFOREST per year as result of a grid search with <code>beam=1</code> , <code>segment_score=max</code>	40
6.4	A general set of parameters for iFOREST.	40

6.5	Results of the feature selection experiment. The progress of the algorithm is shown in Figure 6.6.	44
6.6	Live evaluation: set-up for 2017 and 2018.	49
6.7	Confusion matrices and metrics of the results after live evaluation on <i>all features</i> with <code>anomaly_detector=GMM</code> , <code>beam=1</code> , <code>segment_score=max</code> , <code>threshold=99%</code> . Hyperparameters are set per year to the best results of the grid search (see Table 6.1).	50
6.8	Confusion matrices and metrics of the results after live evaluation on <i>all features</i> with <code>anomaly_detector=GMM</code> , <code>beam=1</code> , <code>segment_score=max</code> , <code>threshold=95%</code> . Hyperparameters are set per year to the best results of the grid search (see Table 6.1).	50
6.9	Confusion matrices and metrics of the results after live evaluation on the optimal <i>feature subset</i> (derived in Section 6.3) with <code>anomaly_detector=GMM</code> , <code>beam=1</code> , <code>segment_score=max</code> , <code>threshold=99%</code>	51
6.10	Confusion matrices and metrics of the results after live evaluation on the optimal <i>feature subset</i> (derived in Section 6.3) with <code>anomaly_detector=GMM</code> , <code>beam=1</code> , <code>segment_score=max</code> , <code>threshold=95%</code>	51
6.11	Attributes of a segment in COBRAS.	53
6.12	Confusion matrix and metrics using the pipeline with parameters of Table 6.2, <code>anomaly_detector=GMM</code> , <code>threshold=99%</code>	58
6.13	Confusion matrix and metrics using the pipeline with parameters of Table 6.4, <code>anomaly_detector=IFOREST</code> , <code>threshold=95%</code>	59

List of Abbreviations and Symbols

Abbreviations

ABT	Accelerator Beam Transfer
AUC	Area Under the Precision-Recall Curve
CALS	CERN Accelerator Logging Service
CERN	European Organization for Nuclear Research
FN	False Negative
FP	False Positive
GMM	Gaussian Mixture Model
iFOREST	Isolation Forest
IPOC	Internal Post Operational Check
KISS	Kicker Injection Soft Start
LHC	Large Hadron Collider
MKI	Kicker Injection Magnets
NXCALS	Next Generation Accelerator Logging Service
PR	Precision-Recall
SPS	Super Proton Synchrotron
TN	True Negative
TP	True Positive
YETS	Year End Technical Stop

Symbols

A	ampere, unit of electric current
bar	unit of pressure
C	coulomb, unit of electric charge
eV	electronvolt, unit of energy equal to approximately $1.602 \cdot 10^{-19}$ joules
μ s	microsecond
ns	nanosecond
V	volt, unit of electric potential

Chapter 1

Introduction

The Large Hadron Collider (LHC) is the world's largest and most powerful particle accelerator [4]. Maintaining a machine of this size in an efficient and sustainable manner requires some effort. Nowadays, problematic situations (anomalies) during operation are too often detected late in the process by experts who analyse the data. Tracking down what actually caused the anomaly is done manually. It is clear that valuable time is lost during these periods of equipment failure. Furthermore discovering and analysing anomalies is a tedious process. It requires a strong knowledge of the complex relations between the components of the LHC and does not scale for future – larger – accelerators.

This thesis proposes an anomaly detection application which will automatically detect anomalies based on a model of the injection kicker magnets behaviour. Such an application can potentially save a lot of time for CERN engineers. Results of root cause analysis can further guide the engineers to improve the LHC for the future.

Chapter 2 of this work provides all necessary background information. First, an introduction to the operation of the LHC and the injection kicker magnets is given. Next, the anomaly detection problem is formally defined. Common anomaly detectors and evaluation metrics are given. The main principles of feature selection and the clustering algorithm COBRAS are introduced. Finally, the main contributions of previous work on this application by N. Wéry and A. Halilovic are summarized.

Chapter 3 introduces the different types of data which are used for the anomaly detection application. The asynchronous sources make it challenging to combine all kinds of data into a feature vector. A second challenge is to incorporate noisy labels as a ground truth for evaluation.

Chapter 4 defines the problem and motivates the development of an anomaly detection application. Although a substantial amount of work has been invested in this application, there are remaining open problems. First of all, the current application does not detect certain anomalies. This is problematic since CERN wants to be notified when problems occur. It is not clear why the model cannot detect these. Second, it is hard to interpret the output of the anomaly detection application. In previous work, predictions were made for certain time intervals which were not related to the usage of the injection kicker magnets. Third, the visualisation of the

results was quite primitive.

Chapter 5 presents the core of this work. It explains how the open problems stated in the previous chapter are addressed by our application. First, previously undetected anomalies are now detected thanks to feature engineering and fine-tuned raw data filters. Second, new state variables are used to generate predictions that make sense. Predictions are directly linked to the single usage of the machine. Third, a web application is presented which can be used in an interactive way for interpreting and improving the anomaly detector. These three main contributions are combined in a pipeline. It includes feature engineering, selection and a new segmentation method.

Chapter 6 gives the the results of an extensive series of experiments of the anomaly detection application. A comparison is made on different levels. The first level varies the set of features which is used. It demonstrates the impact of selecting an appropriate subset of features. The second level varies the mode of the detector: offline or online. It is the first time that an online experiment is performed.

Chapter 2

Background

In this chapter the necessary background information is introduced. We focus on the CERN LHC, anomaly detection, feature selection and an interactive clustering method called COBRAS. Finally, a summary of the previous work on this research study is given.

2.1 The Large Hadron Collider

The LHC is the largest and most powerful particle accelerator in the world [4]. It was built by CERN, the European Organization for Nuclear Research which is situated in Geneva, near the border of France and Switzerland. Physicists at CERN use the LHC to study advanced properties of particles. One of their missions is to simulate the first moments after the Big Bang and to learn more about the beginning of the universe.

The LHC is the largest machine on the planet [4]. It is a ring with a circumference of 27 kilometres, lying under the ground at a depth of 100 m on average. Inside the ring two beams of particles travel in opposite directions. Using specialised superconducting magnets, the particles are accelerated until they travel at nearly the speed of light. Then they are made to collide. These interactions are studied by physicists.



Figure 2.1: The LHC tunnel (source: [15])

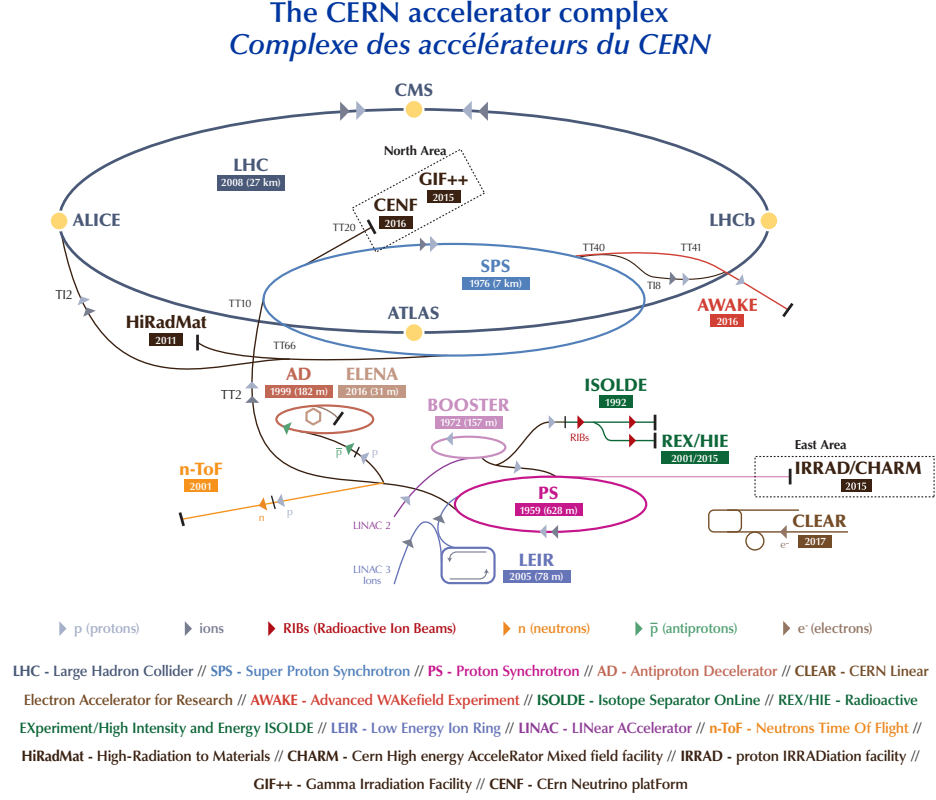


Figure 2.2: The CERN accelerator complex (source: [14])

2.1.1 The accelerator complex

Particle accelerators in general have a limited dynamic range to increase the energy of the particles. To achieve the levels needed to reach a speed close to the speed of light, a chain of accelerators is needed. At the beginning, particles are injected into a first accelerator which increases the energy up to the maximum level it can achieve. Then the particles are extracted from the current accelerator and injected into the next one. By iterating this process, the beam of particles travels through a sequence of machines. It continues until the particles are injected from the Super Proton Synchrotron (SPS) into the LHC. Here, the energy level is increased for a last time up to the maximum energy level of 6.5 TeV. Then the beams are made to collide. An overview of the CERN accelerator complex is shown in Figure 2.2.

2.1.2 Injection Kicker Magnets

Within the accelerator complex, particle beams are extracted from one accelerator and injected into another one. The hardware responsible for this action is called *beam transfer equipment*. Within CERN, it is the responsibility of the Accelerator

Table 2.1: MKI magnet installation names

Location	Installation name	Beam
UA23	MKI2	B1
UA87	MKI8	B2

Beam Transfer (ABT) group to design and maintain this equipment. Critical for LHC operation are the Injection Kicker Magnets (MKI) [2].

A MKI installation consists of four magnets, which are named A, B, C and D. Each pair of magnets (A and B, C and D) is powered by one generator, which has one Resonant Charging Power Supply (RCPS) and two Pulse Forming Networks (PFNs); thus one PFN per magnet.

The LHC requires two injection installations (named MKI2 and MKI8), one for each counter-rotating beam. This brings the total of magnets to 8. Each system of MKI magnets has a certain location and installation name, see Table 2.1. They will be referred to in the variable names of the dataset.

2.1.3 Kicker injection soft start conditioning

As described above, the MKI magnets' function is to inject particles into the LHC. This process takes a relative short time, whereas the experiment inside the LHC goes on for several hours.

During a physics run the magnets cannot be pulsed, since this would influence the ongoing operation. However since the magnets require a high voltage level, de-conditioning might occur during that time. When a magnet at a low pressure level (vacuum) is pulsed in this condition with high voltage, it might cause a vacuum breakdown (i.e. sparking inside the magnet).

To avoid this problem, a Kicker Injection Soft Start (KISS) phase [1] has been added. It slowly increments the voltage level in a safe way until the nominal voltage is reached. The KISS conditioning is executed in parallel with a start-up phase for setting machine parameters before each beam injection. It is relevant for this work because it will generate a very specific pattern in the data which is introduced in Chapter 3.

2.1.4 LHC schedule

The usage of the LHC is scheduled beforehand for a duration of one year. Periods of normal usage are interleaved with machine development and planned technical stops. Every year there is a Year End Technical Stop (YETS) starting in December and continuing until the beginning of March. When modelling normal behaviour of the machine, data generated during maintenance or test periods should be excluded.

2.1.5 CERN accelerator logging service

During LHC runs a vast amount of operational machine data is generated. To store all data from these devices, the CERN Accelerator Logging Service (CALS) [3] is used. It aims to centralise all information and to avoid duplicating the data logging effort.

CALS was originally developed in 2006 with an infrastructure of two Oracle databases. It was designed for a data volume of 5 TB per year [3]. However in 2017 the throughput was 1.2 TB per day. This dramatic increase of data load has led to performance issues for user queries. J. Wozniak (CERN) mentioned at the Spark Summit in 2017 that it took 12 hours to retrieve 24 hours of data. This situation is clearly not desirable.

Therefore a next generation logging service (NXCALS) [10] is currently developed. It makes use of Hadoop and Spark clusters and enables more efficient analysis of bigger data sets over a longer period of time.

2.2 Anomaly detection

Anomaly detection (or *outlier detection*) is the task of detecting abnormal patterns in data which are called *anomalies* (or *outliers*). It can be interpreted as a binary classification task: classify an example as anomalous or not. There is a wide range of applications in anomaly detection: card fraud detection, malignant tumor detection, etc. We formally define anomaly detection as a mapping from input to output:

- **Input:** a set of input vectors with values for certain features
- **Output:** the set of input vectors with an additional value per feature: an anomaly score $\in [0, 1]$

By convention a high score indicates a high probability of being anomalous and vice versa. Anomalous examples will be referred to as the *positive* examples, whereas normal examples are referred to as *negative*.

Using the output of an anomaly detector, a *binary prediction* (anomalous or not) can be made for each input vector. Choose a threshold t and predict the vector as anomalous if its score is higher than t , otherwise predict the vector as not anomalous.

2.2.1 Unsupervised learning

Anomaly detection is usually applied in the setting of unsupervised learning. This means that training examples are not labelled, but assumed to be normal (negative). When the number of anomalies compared to the total number of examples is rather small, it is reasonable to make this assumption. This approach is also used in this work.

2.2.2 Interpretability

Some anomaly detectors cannot only detect anomalies but also offer possibilities to track down the cause of a predicted anomaly. We refer to this property as *interpretability*.

Interpretability is especially useful for the CERN application. Since the dataset consists of 120 variables, it would be an important benefit to know which variable or even subset of variables caused this anomaly.

2.2.3 Techniques

Different techniques have been applied to the problem of anomaly detection. A structured overview of all techniques is given in the survey of V. Chandola [5]. This work focusses on two techniques called Gaussian Mixture Model and Isolation Forest.

Gaussian Mixture Model

A Gaussian Mixture Model (GMM) assumes that each of the given data points belong to one of k Gaussian clusters. These clusters are determined using an Expectation-Maximization algorithm. It is a natural choice to assign a data point a high anomaly score, if it has a low probability to belong to any of the k clusters, and vice versa.

The performance of a GMM is influenced by the number of components. When the number of components is too small, useful relations within the data might not be learnt. On the other hand, when there are too many components overfitting will occur. A theoretical method to determine this parameter is the Bayesian Information Criterion (BIC) [17]. The model with the lowest BIC score is usually optimal. However, Wéry has followed this approach [21] but concluded that for two or more components the BIC score was constant.

A disadvantage of GMM is the lack of interpretation. There is no clear connection between certain variable values and a predicted anomaly.

Isolation Forest

In contrast to the previous method Isolation Forest (iFOREST) does not model normal behaviour but explicitly isolates anomalies. This idea was proposed by Liu, Ting and Zhou [11] in 2008 and is implemented with an ensemble of decision trees. At every node of the decision tree, a set of instances is partitioned in a random fashion. The process is continued until all instances are isolated.

Isolation Forest assumes that anomalies are easier to isolate and hence will lead to a shorter path length. The more the attributes of the data distinguish between normal instances and anomalies, the easier this process is. A visualisation is shown in Figure 2.3. Isolating the normal point x_i requires a higher number of splits, whereas isolating the anomalous point x_o requires a lower number of splits.

A main advantage of Isolation Forest is the support for interpretability. One can easily observe a single decision tree of the ensemble and inspect the nodes. It is exactly known why an instance was predicted as anomalous. By combining this

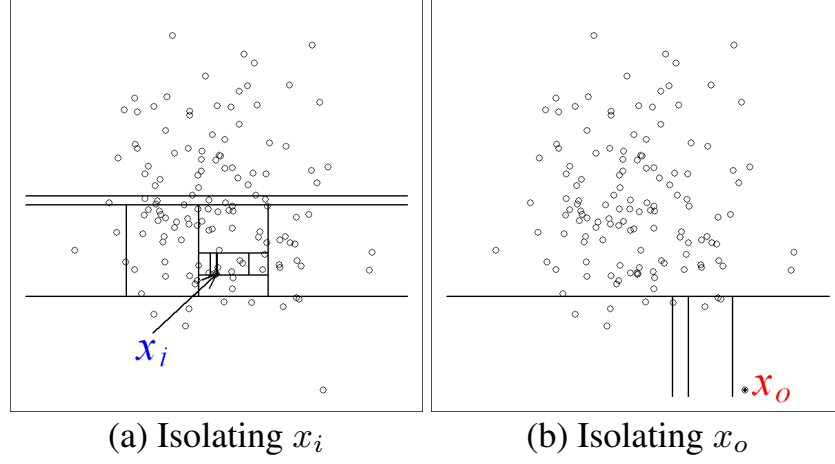
Figure 2.3: Isolating x_i and x_o (source: [11])

Table 2.2: Confusion matrix

	actually positive	actually negative
predicted as positive	TP	FP
predicted as negative	FN	TN

information of all trees in the ensemble, one can get a certain intuition about the cause of a predicted anomaly.

2.2.4 Evaluation metrics

Formal evaluation metrics are necessary to evaluate an anomaly detection application. They can also be used to compare the performance of different detectors.

For an evaluation metric, it is assumed that the ground truth of each example is known. By choosing a threshold t and assigning labels as described above, we can now compare the prediction to the ground truth. The result is summarized in a so-called confusion matrix. It contains four classes: true positive (TP), false positive (FP), false negative (FN) and true negative (TN). They are defined in Table 2.2.

Precision and recall

The quality of the output is often expressed in terms of precision and recall. Precision indicates how many of the examples that were predicted anomalous are actually anomalous. Recall indicates how many of the actual anomalies were predicted as anomalous.

$$\text{precision} = \frac{TP}{TP + FP} \quad (2.1)$$

$$\text{recall} = \frac{TP}{TP + FN} \quad (2.2)$$

In a more intuitive way: high precision means that predictions of anomalies are most of the time an actual anomaly, while high recall means that most of the anomalies were actually found with the detector.

Area under the Precision-Recall Curve

Every threshold yields a value for both precision and recall. The combinations form a curve which is called the Precision-Recall Curve (PR Curve). The area under the PR curve (AUC) is a good indication for the quality of the detector. An ideal detector has an AUC of 1.

Average anomaly rank

Another metric is the average anomaly rank (**rank**). Let l be the list of predictions, sorted by decreasing anomaly score. List l contains elements $l_0, \dots, l_n$ where $\text{score}(l_i) > \text{score}(l_{i+1})$ for $i \in \{0, \dots, n-1\}$. We define the average anomaly rank as:

$$\text{rank} = \frac{\sum_{l_i \text{ is actual anomaly}} i}{TP + FN} \quad (2.3)$$

where the denominator denotes the number of anomalies.

Suppose there are four instances to predict, of which three are anomalous (A) and one is normal (N). A first perfect detector would assign the highest anomaly scores to the anomalous examples. A second detector however might incorrectly assign a higher anomaly score to a normal example. These orderings together with the resulting rank are shown in Table 2.3. It is clear that a rank of 1 represents a perfect detector. The higher the rank, the worse the anomaly detector.

Table 2.3: Computing rank

Element	Detector 1	Detector 2
l_0	A	A
l_1	A	N
l_2	A	A
l_3	N	A
rank	$\frac{0 + 1 + 2}{3} = 1$	$\frac{0 + 2 + 3}{3} = 1.67$

2.3 Feature selection

Feature selection is the process of selecting a subset of features which are then used to perform a task. In our case this is the training and evaluation of an anomaly detector. Selecting an optimal subset is a preprocessing step which reduces the data dimensionality. It should lead to a better result, be computationally less intense and yield a higher interpretability [12].

Popular feature selection algorithms have been analysed by K. Sutha [7]. This work has led to a classification of the algorithms into three categories: Filter, Wrapper and Hybrid Method.

The *Filter Method* selects the feature subset based on the data itself. It is independent of the task for which the data is used. As a consequence it is very general. The *Wrapper Method* selects the feature subset based on its predicted performance on the task. This is usually a computationally more intense method for large datasets, yielding better results. The *Hybrid Method* combines the previous approaches. It will first perform a Filter Method and then a Wrapper Method. It is a trade-off between the low computational intensity of the Filter and the higher quality result of the Wrapper Method.

2.4 COBRAS: interactive clustering with super-instances

An anomaly detection application assigns anomaly scores to instances, leading to a binary prediction. Usually these results are reported to a team of experts. However, some types of anomalies might not be of interest to the experts. It might be a normal situation where equipment is replaced causing an anomaly. This is a false alarm which should not be examined. Therefore it is beneficial to incorporate expert feedback. The feedback could be: ignore a certain type of “intervention” anomalies. In this work a method to achieve this is proposed using the interactive clustering algorithm COBRAS [18].

Clustering is the task of classifying a set of input data into multiple clusters. The motivation to use COBRAS is that clustering is an inherently subjective task. The desired clustering does not only depend on the given dataset, but also on the goal for which it is used. Therefore user interaction is required to obtain the target clustering.

In the case of COBRAS, user interaction is provided in the format of pairwise queries. Given two instances, the system queries the user with the question: “*Should these 2 instances belong to the same cluster?*”. The answer (yes or no) results into a *must-link* or *cannot-link* constraint. A *must-link* constraint means that the instances must belong to the same cluster, whereas a *cannot-link* constraint expresses that they cannot belong to the same cluster.

Central in the work of COBRAS is the concept of a *super-instance*. Van Craendonck defines it as “*a local region in the data in which all instances are assumed to belong to the same cluster*” [18]. It gives rise to a hierarchy of clusters. A clustering

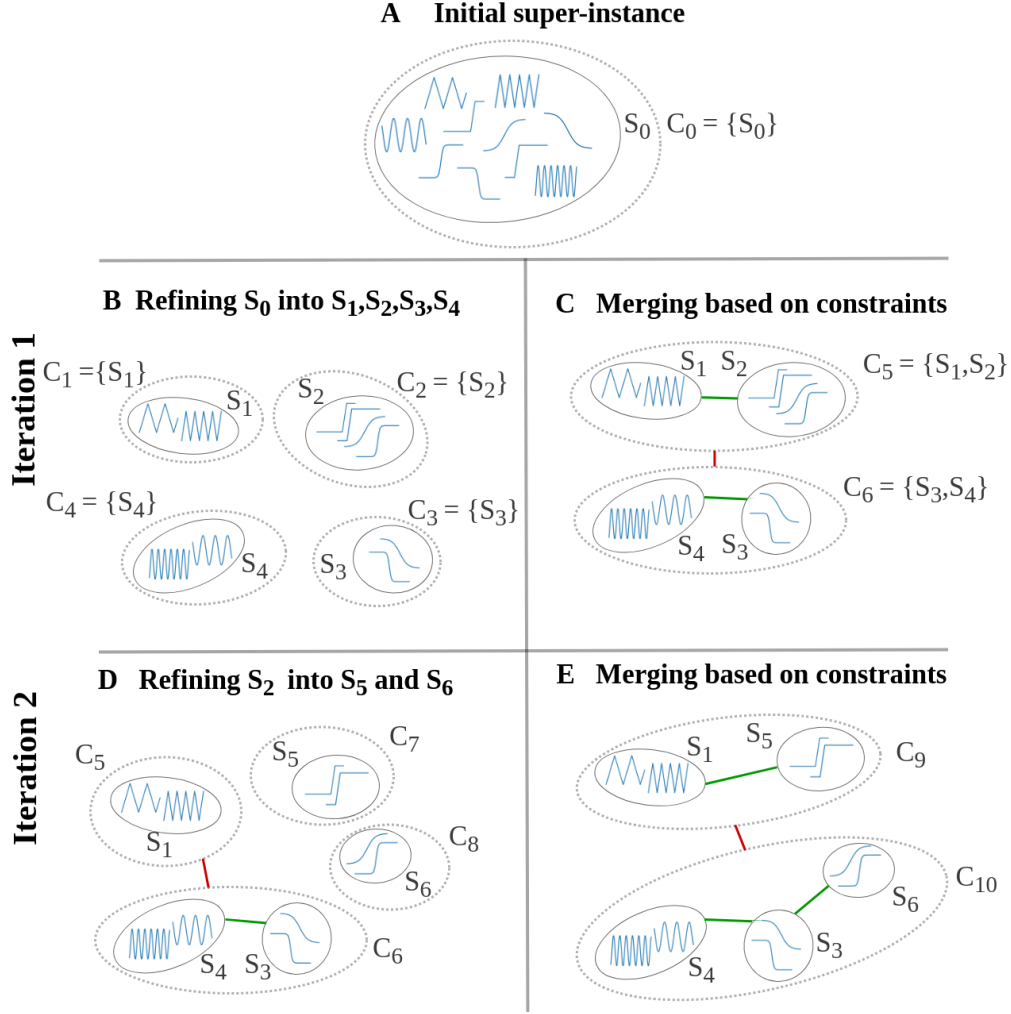


Figure 2.4: Constraint-based clustering with COBRAS (source: [19])

is a set of clusters, and a cluster consists of set of super-instances. A super-instance consists of a set of instances.

Given the concept of constraints and super-instances, COBRAS builds a clustering in an iterative process. Each iteration consists of two steps: refine and merge. The first step refines the super-instance into smaller super-instances based on K-means. The second step merges the super-instances using the input of the user in the form of constraints. An example is shown in Figure 2.4. This iterative process ends with a clustering when the so-called *query budget* has been spent. It limits the number of questions to be answered by the expert.

2.5 Previous work

Niels Wéry [21] and Armin Halilovic [8] have previously worked on an anomaly detection application for the MKI magnets at CERN. This section discusses the most important contributions of their work.

Wéry was the first one to work on this topic. He designed a Mongo database abstraction for querying the data provided by CERN. It is still being used. Wéry trained a model using GMM on a feature dataset consisting of continuous data, IPOC data and sliding window features. The results were evaluated using the concept of anomalous segments. Such a segment was quite arbitrary since it was determined by a constant parameter and not linked to the usage of the machine. Wéry suggested to use other anomaly detection methods, to improve the interpretability of the model.

Halilovic continued the work of Wéry. His main contribution is an anomaly detection pipeline consisting of four steps: preprocessing, anomaly detection, postprocessing and evaluation. The preprocessing step transforms the data into features that can be used for training a model. There are three transformations. First, it filters out wrong sensor measurements, as provided by CERN. Second, it computes sliding window features of continuous data. Third, features are scaled before they are passed to the training phase. The anomaly detection step trains and scores the given features. Halilovic proposed using Isolation Forest as an anomaly detector and compared the results with GMM and a few other dummy detectors. The postprocessing step then segmented the examples. This segmentation is based on a fixed parameter. Each segment is given a ground truth and predicted score. The evaluation step is the last step of the pipeline. Its input consists of segments with a ground truth and a predicted anomaly score. Using a threshold, the predicted scores were converted to predictions. Standard evaluation metrics were used to evaluate the results.

Halilovic concluded that GMM performed better than Isolation Forest on the data of 2015-2016. Surprisingly the Isolation Forest on the entire feature set performed worse than a dummy detector. Then, he used a feature subset consisting of the IPOC data for training and testing Isolation Forest. This experiment resulted in a better prediction than the dummy detectors but was still worse than GMM. It gave the intuition that using a feature subset could help Isolation Forest to model anomalies, with the benefit of interpretability.

Chapter 3

Data

The data provided by CERN is vast and complex. It is generated by different asynchronous sources which were not set up to be used in machine learning. This makes it challenging to merge all the data into a feature vector. In this chapter the relevant information about the used dataset is given.

3.1 General information

The data provided by CERN has evolved over time. Whereas previous research studies used data from June 2015 until April 2016, this research study focusses on more recent data. An important motivation to do this is the introduction of state data in March 2016. This data makes it possible to use a more intuitive segmentation method (see Section 5.3). Furthermore the beam transfer equipment of the LHC machine has been upgraded over time and studying more recent data is a reasonable choice. An overview is shown in Table 3.1.

3.2 Types of data

The dataset contains both continuous and discrete variables. Some of them are time series, others are not. The sampling method also varies from one to another. It could be fixed or based on a trigger.

Table 3.1: The time span of used data

Iteration of the research study	start	end
Thesis 1 by Wéry [21]	22 June 2015	12 April 2016
Thesis 2 by Halilovic [8]	22 June 2015	9 September 2016
Current work	10 March 2016	1 January 2019

Table 3.2: Continuous data

Variable	Description	Unit	Data type
PRESSURE	Vacuum level	mbar	NUMERIC
TEMP_MAGNET_UP	Upstream temperature	°C	NUMERIC
TEMP_MAGNET_DOWN	Downstream temperature	°C	NUMERIC
TEMP_TUBE_UP	Tube upstream temperature	°C	NUMERIC
TEMP_TUBE_DOWN	Tube downstream temperature	°C	NUMERIC

3.2.1 Continuous data

The first type of data is continuous. Examples are the pressure and temperature. The continuous variables are shown in Table 3.2.

The temperature is being measured *upstream* and *downstream* with reference to the beam. The upstream temperature is measured where the particles are entering the magnet and the downstream temperature where they leave the magnet.

The continuous data is sampled at a fixed frequency of 2 Hz. Since storing all measurements is not efficient, CALS values are often only stored upon change, or if it did not change for a long time. This last parameter is configured by the CERN user and ranges from 15 minutes to one day.

The 5 continuous variables are measured for each magnet. Since there are 4 magnets (A, B, C and D) per system (MKI2 and MKI8), there are 40 continuous variables in total.

Name convention

MKI.A5L2.B1:TEMP_MAGNET_UP is an example of a fully specified name in the database. We can decompose it as follows: it refers to data of the MKI magnets, more specifically magnet A, for beam 1 (i.e. system MKI2) and the values are the upstream temperature. Other variables names can be interpreted in a similar way.

3.2.2 IPOC data

The second type of data is Internal Post Operational Check (IPOC) data [13]. Examples are the kick strength and length of the magnet current pulse. The IPOC variables are shown in Table 3.3.

In contrast to continuous data, the IPOC data is not sampled at a fixed frequency. It is sampled with *acquisition trigger sampling* which means that when a certain trigger occurs, the digitizers capture the data from their sensors at the same time. Such a trigger corresponds to the usage of the magnets (i.e. fast pulsed current to deflect the beam into the LHC) and therefore these timestamps are useful in understanding the behaviour of the machine.

The IPOC data used here is really the resulting calculations from a voltage or current waveform analysis. An industrial PC runs an analysis algorithm and the properties of the waveform are extracted as IPOC data.

Table 3.3: IPOC data

Variable	Description	Unit	Data type
E_KICK	LHC energy at injection time	GeV	NUMERIC
I_STRENGTH	Kick strength (Flat-Top average)	kA	NUMERIC
T_DELAY	Kick delay (Start point time - Trigger time)	μ s	NUMERIC
T_FALLTIME	Kick fall time (EndThreshold point time - End point time)	μ s	NUMERIC
T_LENGTH	Kick length (End point time - Start point time)	μ s	NUMERIC
T_RISETIME	Kick rise time (Start point time - StartThreshold point time)	μ s	NUMERIC
T_START_TH	Time of StartThreshold point	μ s	NUMERIC

Table 3.4: State data

Variable	Description	Unit	Data type
CONTROL	Pulse generator control	–	TEXTUAL
MODE	Pulse generator mode	–	TEXTUAL
SOFTSTARTSTATE	Soft-start state	–	TEXTUAL
STATUS	Pulse generator status	–	TEXTUAL

The 7 IPOC variables are stored for each magnet. Since there are 4 magnets (A,B, C and D) per system (MKI2 and MKI8), there are 56 IPOC variables in total.

3.2.3 State data

The third type of data refers to the state of the MKI magnets and origins from a high-level surveying Programmable Logic Controller (PLC). It is categorical data and for each of the variables in Table 3.4, the possible values are shown in Table 3.5.

Especially useful are the STATE:CONTROL and STATE:MODE variables. The first one indicates whether the control room at CERN has the control over the magnets (remote), or whether it was used for research or testing by the equipment group (local). The second variable gives information about the actual usage of the machine. When this value changes from ON to any other value, it indicates a finished injection cycle (i.e. magnet usage).

The state data was added to the logging database on the March 10, 2016. It has not been used in previous work, but will be used in this thesis.

The 4 variables are stored for each system (MKI2 and MKI8), so there are 8 state variables in total.

3.2.4 LHC data

The fourth type of data gives information about the beam intensity and the length of bunches in the LHC. It is not specific to the MKI magnets but gives general information about the LHC beam; there is a direct link with the magnets however since for example shorter bunch spacing leads to increased temperatures. A summary is shown in Table 3.6.

Table 3.5: Values of state data

(a) Values of pulse generator control	(b) Pulse generator mode																						
<table> <tr><th colspan="2">STATE:CONTROL</th></tr> <tr><td>5</td><td>REMOTE</td></tr> <tr><td>6</td><td>LOCAL</td></tr> <tr><td>7</td><td>SOFTSTART</td></tr> </table>	STATE:CONTROL		5	REMOTE	6	LOCAL	7	SOFTSTART	<table> <tr><th colspan="2">STATE:MODE</th></tr> <tr><td>0</td><td>UNKNOWN</td></tr> <tr><td>1</td><td>ON</td></tr> <tr><td>2</td><td>OFF</td></tr> <tr><td>3</td><td>STANDBY</td></tr> <tr><td>4</td><td>FAULTY</td></tr> <tr><td>5</td><td>WARMING_UP</td></tr> </table>	STATE:MODE		0	UNKNOWN	1	ON	2	OFF	3	STANDBY	4	FAULTY	5	WARMING_UP
STATE:CONTROL																							
5	REMOTE																						
6	LOCAL																						
7	SOFTSTART																						
STATE:MODE																							
0	UNKNOWN																						
1	ON																						
2	OFF																						
3	STANDBY																						
4	FAULTY																						
5	WARMING_UP																						
(c) Values of soft-start state	(d) Pulse generator status																						
<table> <tr><th colspan="2">STATE:SOFTSTARTSTATE</th></tr> <tr><td>0</td><td>UNKNOWN</td></tr> <tr><td>1</td><td>ON</td></tr> <tr><td>2</td><td>OFF</td></tr> <tr><td>3</td><td>BUSY</td></tr> </table>	STATE:SOFTSTARTSTATE		0	UNKNOWN	1	ON	2	OFF	3	BUSY	<table> <tr><th colspan="2">STATE:STATUS</th></tr> <tr><td>0</td><td>UNKNOWN</td></tr> <tr><td>1</td><td>OK</td></tr> <tr><td>2</td><td>WARNING</td></tr> <tr><td>3</td><td>ERROR</td></tr> </table>	STATE:STATUS		0	UNKNOWN	1	OK	2	WARNING	3	ERROR		
STATE:SOFTSTARTSTATE																							
0	UNKNOWN																						
1	ON																						
2	OFF																						
3	BUSY																						
STATE:STATUS																							
0	UNKNOWN																						
1	OK																						
2	WARNING																						
3	ERROR																						

Table 3.6: LHC data

Variable	Description	Unit	Data type
BEAM_INTENSITY	Beam intensity	C	NUMERIC
BUNCH_LENGTH_MEAN	Mean measured bunch length	s	NUMERIC

The variable BEAM_INTENSITY expresses the total charge of the particles in the beam. If it is greater than zero, it is sure that the MKI magnets have been used to inject particles. However, the contrary does not necessarily hold.

The two variables are stored for both beam 1 and 2, which results in four LHC variables.

3.2.5 Timing controller data

The fifth type of data concerns the timing settings of the MKI magnets. These configuration parameters determine the amplitude and length of a pulse. An overview is shown in Table 3.7.

Most variables have a vector type. This is a relic from another data acquisition system. For the MKI magnets, only the first value of the vector must be used.

Controller data is given for each system (MKI2 and MKI8), so there are 12 variables in total.

3.3 Logbook

CERN has provided additional information in the format of an electronic logbook (e-logbook). It can be used to identify whether an anomaly indicated by our application

Table 3.7: Controller data

Variable	Description	Unit	Data type
KICK_COUNT_TO_PLAY	Kick count to play	–	NUMERIC
KICK_DELAY_TO_PLAY	Kick delay to play	ns	VECTOR NUMERIC
KICK_ENABLE_TO_PLAY	Kick enable to play	–	VECTOR NUMERIC
KICK_LENGTH_TO_PLAY	Kick length to play	ns	VECTOR NUMERIC
KICK_STRENGTH_TO_PLAY	Kick strength to play	V	VECTOR NUMERIC
KICK_TIME_TO_PLAY	Kick time to play	ns	VECTOR NUMERIC

Table 3.8: A logbook entry

Key	Value
id	2336542
timestamp	2017-05-06 05:27:00
username	mbarnes
tag	anomaly
beam	2
comment	Called by CCC at 7.07hrs for interlock on MKI8 generator 2. Process 2 Step 6 showed an FSC interlock but nothing particular on FSC screen. IPOC showed no current in MS and TDR 8D conducting early (MKI8D DS erratic): reservoir voltage looks stable. To be investigated by EC why nothing showed up on Fast Interlock screen.

actually was indicated as an anomaly by experts. An example entry is shown in Table 3.8.

The entries in the logbook were analysed by a CERN expert who assigned one of the following labels depending on the circumstances:

- Anomaly: the equipment does not behave as expected and this behaviour should be detected as an anomaly
- Fault: the equipment does not behave as expected, but this behaviour is not relevant for our use case or it cannot be observed in the provided data sets
- Info: information useful for CERN experts
- Intervention: changes made to the hardware or software
- Research: certain conditions were created for a research purpose (e.g. machine development)

It is important to keep in mind that each logbook entry was created by hand and that some events or problems might not have been logged. Maybe an event was added late, or not discovered at all. To capture this uncertainty partially, the timestamps of the logs are assumed to have an uncertainty range of 12 hours (determined in consultation with CERN experts). For example if an anomaly was logged at 11 AM

Table 3.9: The amount of labels

(a) Beam 1

Year	Anomaly	Fault	Info	Intervention	Research
2016	21	11	47	22	1
2017	10	6	29	16	0
2018	12	9	37	25	0
Σ	43	26	113	63	1

(b) Beam 2

Year	Anomaly	Fault	Info	Intervention	Research
2016	7	11	26	17	2
2017	7	8	31	18	0
2018	8	6	30	14	0
Σ	22	25	87	49	2

and the detector indicates an anomaly at 6 AM, then this will be seen as a correct detection since it is within the uncertainty range.

To interpret the results of predictions later in this work, it is interesting to have an idea about the actual amount of anomalies occurring. Table 3.9 shows the amount of labels for each beam per year.

3.4 Summary

In this chapter, the 120 variables coming from asynchronous sources were introduced. The complete list can be found in Appendix A. Together with the noisy labels in the logbook, this data is the input of the anomaly detection application.

Chapter 4

Problem statement

4.1 Problem

The LHC is currently the world's largest and most powerful particle accelerator [4]; limited resources require using it in an efficient and sustainable manner though. Taken into account the power of the circulating beams at full energy [6], a situation where problems inside the machine are detected too late is clearly not desirable. It would also be beneficial to track down the source of problems in order to optimise the machine for later usage.

However, due to its size and complexity this task is not at all obvious. Nowadays, problematic situations (or anomalies) are often detected only afterwards by experts who analyse the data. Root cause analysis of anomalies is also done by hand. It is a tedious process that requires a strong knowledge of the complex relations between the components of the LHC. Furthermore for future accelerators this process will no longer be possible for humans due to the increased complexity.

The problem is summarized as follows:

The manual detection of anomalies in the LHC operational machine data is a tedious process that does not scale.

4.2 Goal

In response to this problem, this thesis proposes an anomaly detection application which can discover anomalies automatically. Such an application could reduce machine downtime and save many hours spent by employees at CERN analysing data. It could also reveal relationships between the data which were not known before.

We distinguish two variants of the problems to be solved by this application. The first variant is an offline detector which detects anomalies based on historical data. The second variant is an online detector. It is to be used during normal machine operation, using live data. It can be run continuously to detect anomalies as they

occur. This is a more challenging case than the first one but it gives more value to the application as it will provide operation with “alarms” during experiments.

Since modelling all operational data of the LHC at once would be infeasible, a data subset was selected. A critical machine component is the beam transfer kicker magnet with high-voltage pulse generator, responsible for transferring bunches from one accelerator into the next one. The magnets used for transferring particles from the SPS into the LHC are called the injection kicker magnets (MKI). Since they have a critical impact on the functioning of the entire complex, we selected this equipment for a first research study.

All elements contribute to the following goal of this work:

The development of an anomaly detection application for the injection kicker magnets of the Large Hadron Collider which can operate in an offline and online mode.

The next chapters describe how this goal was accomplished.

Chapter 5

The anomaly detection application

Several iterations of this research study have yielded an anomaly detection application which is structured as a pipeline. It consists of five steps: preprocessing, anomaly detection, postprocessing, evaluation and visualisation. Each of these steps is explained in this chapter. An overview of the pipeline is shown in Figure 5.1.

5.1 Preprocessing

The first step of the pipeline transforms the raw data from the database into a set of feature vectors. It performs feature engineering, filtering, scaling and finally merges the data. This last task is not trivial because the data is generated by asynchronous sources. Chapter 3 explained that for example IPOC data is collected when the magnets are being used, whereas state data is only collected when the state of a magnet changes.

5.1.1 Feature engineering

It is inherent to anomaly detection to use feature vectors for training and prediction. Although a large absolute value of a data variable might indicate an anomaly, this is not necessarily the case. Possibly both large and small values are normal, but

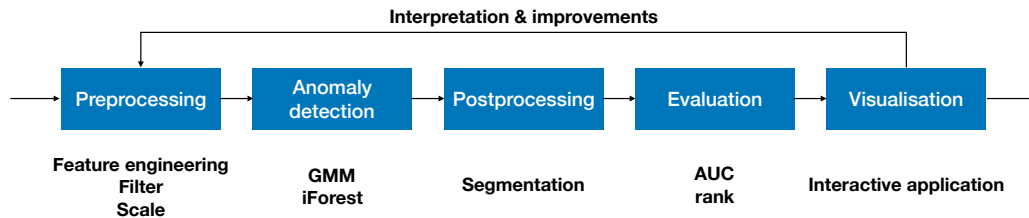


Figure 5.1: The anomaly detection pipeline

only large differences in the data indicate an anomaly. Without the generation of additional features to explicitly describe the temporal behaviour of the data, this type of information could be lost. This motivates temporal features generation. In the literature the process of generating features is called *feature engineering*.

In collaboration with a CERN expert, two types of features were added.

Sliding window features

The first type is sliding window features. They were introduced in the first iteration of this study [21]. We briefly mention the two different types:

- SW_1 : The difference of the value with the moving average, which is based on the N most recent measurements.
- SW_2 : The sum of the N most recent measurements.

Both of these features depend on a sliding window size parameter. In previous work, a sliding window size of 10 minutes was decided in collaboration with CERN. The sliding window features are computed for the continuous and LHC data.

Fourier features

Over time, CERN experts have observed that frequency data of the continuous variables is useful when manually analysing the MKI data. Since this type of information is beneficial for humans, it might also guide the detector to better model the behaviour of the magnets. This observation has led to a second type of temporal features: Fourier features.

To generate Fourier features for a set of input data, the spectrogram of the time signal is computed. This is a representation of the signal which gives an intensity for each frequency within a certain range. To demonstrate the effect of this operation, the pressure variable of magnet D was selected. The values of this variable are shown in Figure 5.2a. The first spike corresponds to an anomaly event in the e-logbook on May 26, 2017. For this specific input, the spectrogram is computed in Figure 5.2b. A detailed view of the low frequencies is also shown in Figure 5.2c since this region has the highest intensity variation.

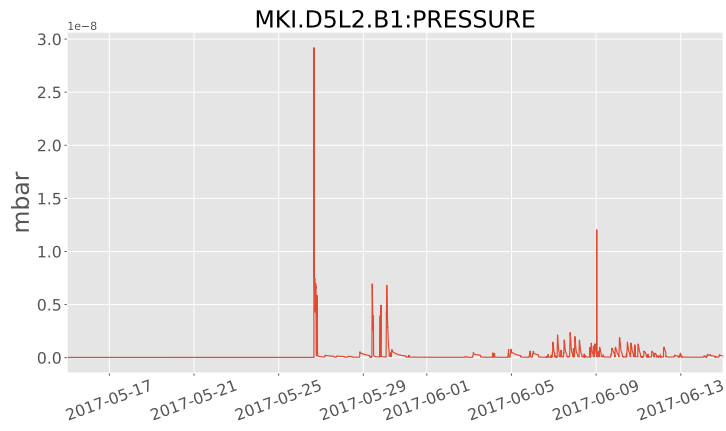
Given these spectrograms both frequency and intensity features are extracted.

Frequency There are two variants of the frequency features:

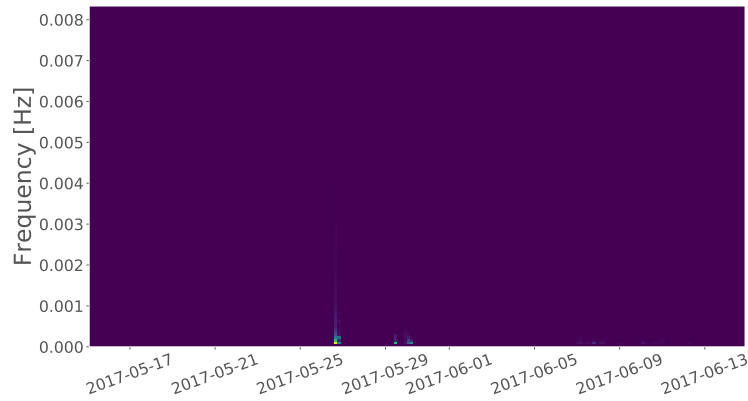
- $Freq_{\max}$ which is the frequency of the maximal intensity at every moment in time.
- $Freq_{i\%}$ which is the maximal frequency among the $i\%$ highest intensities at every moment in time ($i \in \{5, 10, 20\}$).

Intensity There are three variants of the intensity features:

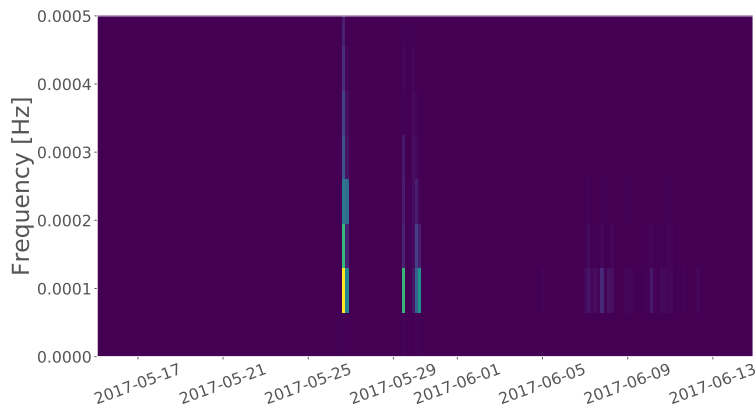
- $Int_{\max variance}$ which is the intensity of the frequency with the highest variance among all frequencies.



(a) Input signal



(b) High level overview of a spectrogram



(c) Detail of a spectrogram

Figure 5.2: Spectrogram: a frequency-intensity representation of a time signal

Table 5.1: The specification of filters removing erroneous sensor measurements.

Variable x	Allowed values	Unit
TEMP_MAGNET	$18 \leq x \leq 60$	°C
TEMP_TUBE	$18 \leq x \leq 120$	°C
PRESSURE	$9 \times 10^{-12} \leq x \leq 5 \times 10^{-9}$	mbar
I_STRENGTH	$x > 1$	kA
T_DELAY	$x > 0.0$	µs
STATE:CONTROL	$x \neq 6$ (i.e. LOCAL)	-

- Int_{maxall} which is the maximal intensity belonging to a frequency at a moment in time.
- Int_{median} which is the median intensity over the considered frequency range at a moment in time.

The output of these features for the pressure of magnet D is shown in Figure 5.3. Notice that for each moment in time the feature $Freq_{20\%}$ is larger than $Freq_{10\%}$ which is larger than $Freq_{5\%}$. This is an expected result since it represents the highest frequency of the top $i\%$ of intensities. The more intensities taken into account, the larger this value can be.

Both frequency and intensity features are computed for all continuous variables (after resampling to 1 minute).

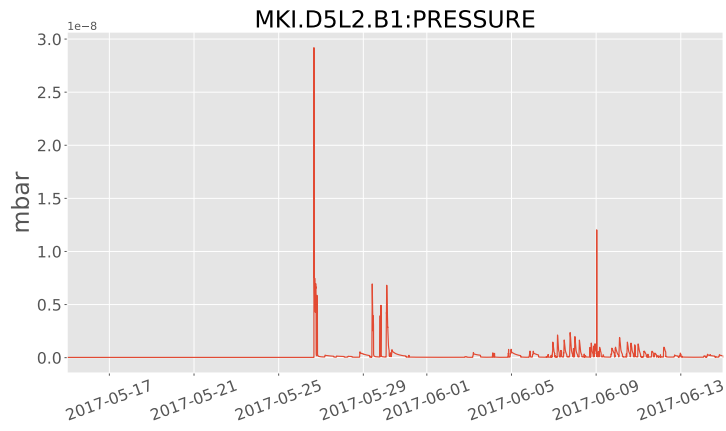
5.1.2 Filter

The set of historical data contains certain values which are physically infeasible. It is quite obvious that for example a negative time value cannot occur. Other examples are caused by for example erroneous temperature sensors. These erroneous measurements have been filtered out. In collaboration with CERN, the filter specifications in Table 5.1 were proposed. Although the filters were also used in previous work, the specification of one filter has changed and an additional filter was added. This was decided after a case study on the predictions for May 1 until August 1, 2017 and has resulted in an improved model.

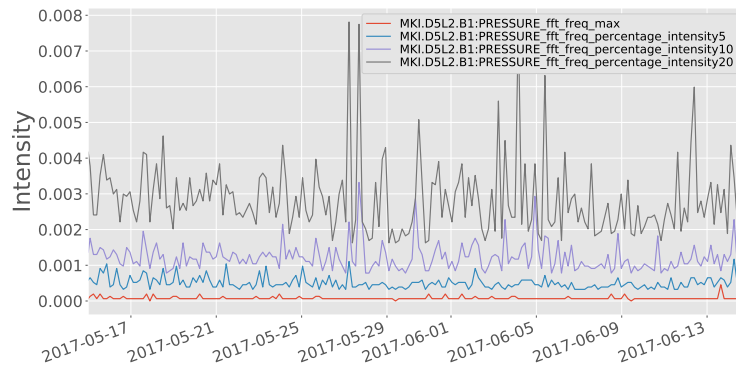
First, the filter for T_DELAY was changed from 10 µs to 0.0 µs. We did this because the filter of 10 µs was too strict to detect certain erratics (pulse patterns which do not follow the normal behaviour). An example is shown in Figure 5.4.

Second, a filter was added to get rid of non-operational data. It ensures that data generated during interventions is not used for training the model. We did this because the model should learn normal behaviour and not special effects occurring when changing the equipment.

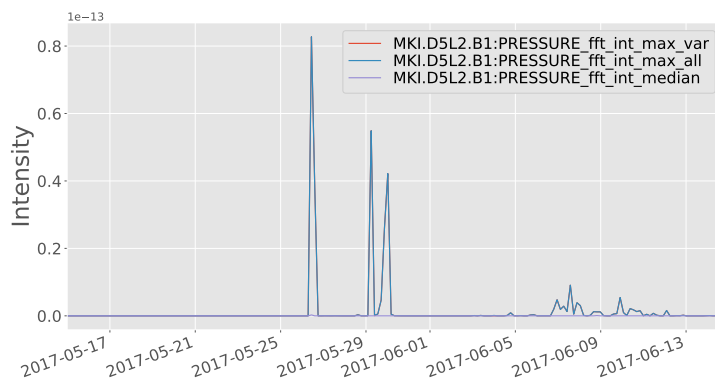
The effect of filtering temperature and pressure data is shown in Figure 5.5. After filtering the pattern of the data is much more interesting to inspect. It shows the continuous variation of the temperature over time.



(a) Input signal



(b) Extracted frequency information



(c) Extracted intensity information

Figure 5.3: Extracted frequency and intensity information

5. THE ANOMALY DETECTION APPLICATION

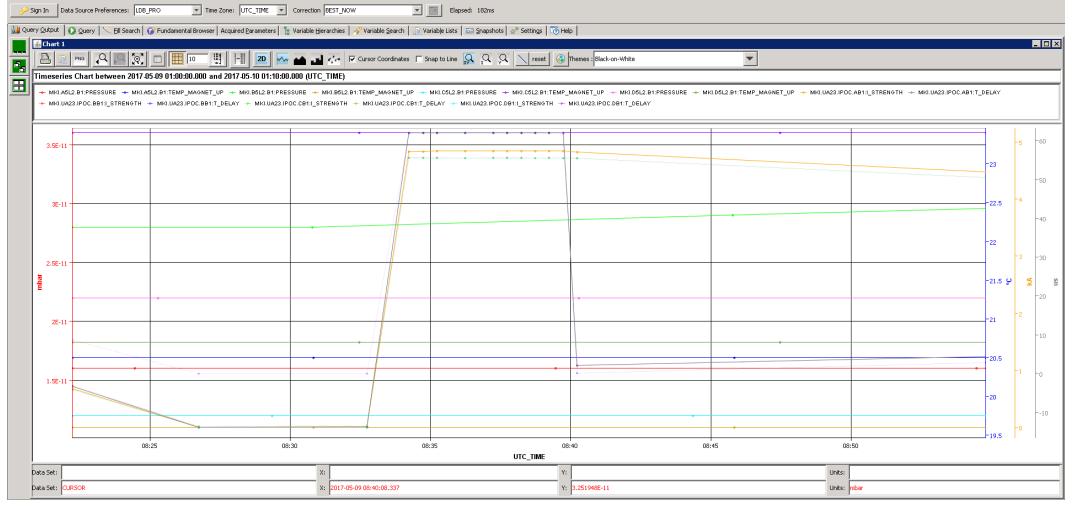


Figure 5.4: False negative due to a wrong filter

5.1.3 Scale

The same scaling method **standard** as in previous work [8] is supported. It normalizes the data by setting the mean to zero and the variance to one:

$$\hat{x} = \frac{x - \mu}{\sigma}, \quad (5.1)$$

where μ is the mean and σ the standard deviation of x . Additionally scaling can be disabled in the pipeline by setting the scaling method to **none**.

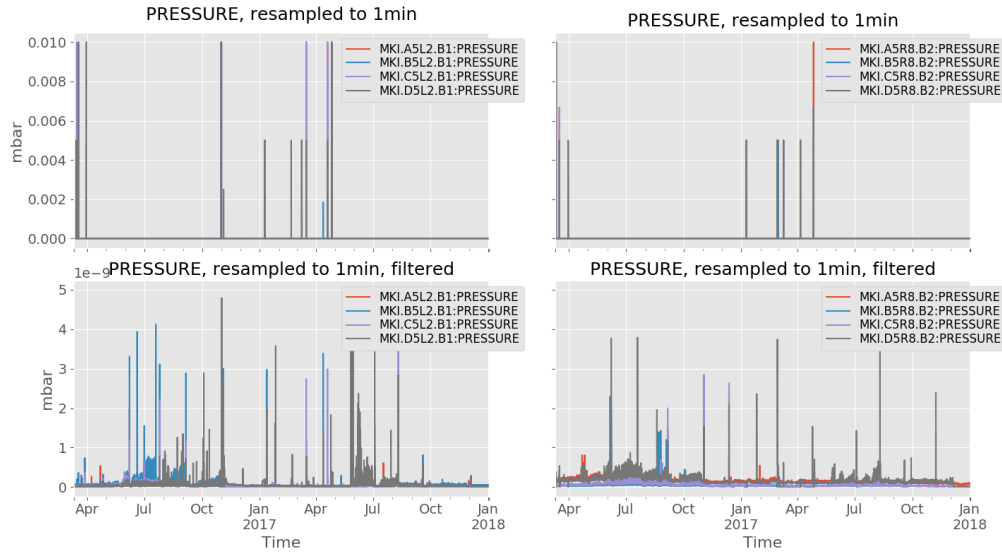
5.1.4 Merging data from asynchronous sources

The MKI data is generated by asynchronous sources. Therefore different types of data usually have different timestamps. However it has been suggested by CERN experts that different kinds of datasets are required to detect all anomalies. Therefore a strategy to combine all data is proposed.

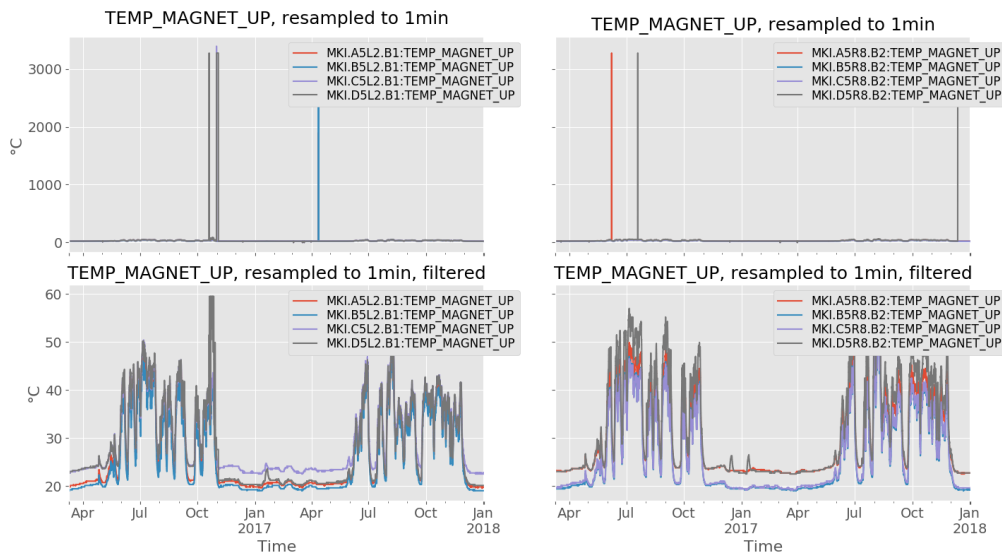
The process starts with the IPOC data. This type of data is only valid at its timestamps and cannot be extrapolated to other moments in time. These IPOC timestamps correspond to the usage of the magnets. As in previous iterations of this study, the idea is to add more data at these timestamps by extrapolating the other data types. Notice that the magnets are used only a tiny fraction of time compared to the complete time of an experiment. However, it is a reasonable choice since the application should detect anomalies in the usage of these machines.

Before joining the other types of data, the IPOC timestamps are rounded up to the second. This process does not cause a loss of information since there is at most one IPOC value per second. Performing rounding has the benefit that it helps to limit the amount of data generated during later resampling of the other data sources.

Next, all other types of data are resampled up to the second and joined with the IPOC data. Only the data at the IPOC timestamps is maintained.



(a) PRESSURE



(b) TEMP_MAGNET_UP

Figure 5.5: The effect of filtering

The result of this process is a set of feature vectors including IPOC, state, controller, continuous and LHC data. It contains sliding window features for the continuous and LHC data, and Fourier features for the continuous data.

5.2 Anomaly detection

The second step takes a set of train and test instances. The anomaly detector is trained on the set of train instances in an unsupervised way. It assumes that all given instances are normal. This assumption is reasonable since the number of anomalies is low. After training, the detector is used to predict the anomaly scores for every test instance.

Any model that predicts anomaly scores can easily be plugged into the framework. So far the focus has been on GMM and iFOREST which were introduced in Section 2.2.3.

5.2.1 Hyperparameters

Hyperparameters are parameters which are set before the training actually begins. They determine the characteristics of the anomaly detector itself and the process of training it. Therefore the actual set of hyperparameters depends on the detector. An optimal choice can experimentally be determined with a grid search (see Section 6.2).

The remainder of this section explains the most important hyperparameters for GMM and iFOREST.

GMM

- `n_components`: integer, the number of Gaussian distributions which are fit on the training data
- `covariance_type`: full, tied, diag or spherical, the type of the covariance matrix of each Gaussian distribution
- `init_params`: kmeans or random, the method which is used to initialise the weights, mean and covariance of each Gaussian distribution
- `n_init`: integer the number of initialisations

iForest

- `n_estimators`: integer, the number of estimators of the ensemble
- `max_samples`: integer, the number of samples from the dataset used to train each estimator of the ensemble
- `max_features`: float, the number of features taken from the dataset to train each estimator of the ensemble (0.9 indicates that 90% of all features are used)

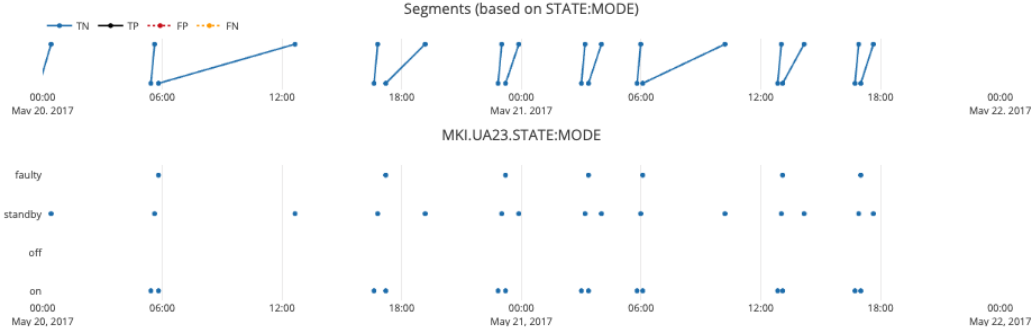


Figure 5.6: Segments based on STATE:MODE data

5.3 Postprocessing

The third step takes a set of scored test instances and transforms these into a set of segments with a score and ground truth. We propose a new method for creating segments.

5.3.1 Motivation

The accelerators at CERN are often used for a certain goal during a set time interval. Anomaly predictions for instances at specific time instants have little meaning for CERN. It is more interesting to extract whether an anomaly occurred during the usage of the magnets in that time interval.

Additionally, this step will make it easier to define an evaluation method in the consecutive step of the pipeline.

5.3.2 Segment

A *segment* is a container for a set of test instances. Instances belonging to the same segment were generated during the same usage period of the magnets.

Segments are created based on the STATE:MODE data (see Section 3.2.3). For each interval starting with ON to any other value a segment is created. The result of this process is shown in Figure 5.6. Tuples are added to the segment to which they belong.

Notice that tuples which do not belong to a segment are ignored. These are rare situations. It is a reasonable choice since under normal circumstances there must be a time interval indicating the usage of the machine.

Segment score

The anomaly score of a segment is defined as a function of the scores of its instances. As in previous work [8], three segment score methods are supported:

- **max**: the worst anomaly score of all instances in the segment

- **top_k**: the average anomaly score of the top 10 worst anomaly scores
- **top_percentage**: the average anomaly score of the top 25% worst anomaly scores

Segment ground truth

To evaluate our predictions, i.e. to verify whether we predicted a segment correctly as anomalous or not, each segments needs a *ground truth*. The labelled e-logbook entries can be used for this purpose. However, these labels are noisy: the actual anomaly might have occurred some hours before or after the e-logbook entry was created. Taking this into account, we define the ground truth of a segment as anomalous if it is within a 12 hours range of an anomaly. Otherwise it is not anomalous. This specific range was motivated in Section 3.3.

Segment pruning

Suppose that an anomaly occurs at 4 PM and there is a segment from 1-2 PM and another segment from 5-6 PM. According to the described method, the ground truth of both segments should be anomalous. If the detector assigns a high anomaly score to both segments, this would be evaluated as a good prediction. However, if only one of them obtains a high anomaly score, it would be evaluated as a bad prediction. This is not desirable since the anomaly which occurred was correctly detected. The uncertainty range of the e-logbook should not influence the evaluation process.

Ideally, we want to take every label of the logbook into account only once for evaluation. Therefore if there are multiple segments with an anomalous ground truth for the same anomaly, they will be merged into a single segment.

5.4 Evaluation

The fourth step has as input a set of segments with their ground truth and predicted anomaly scores. The output is the evaluation of a certain metric. Two metrics are supported.

The first metric is the Area under the Precision-Recall Curve (AUC). This curve is obtained by computing precision and recall for a varying decision threshold. The AUC can be at most 1, and the higher the better.

The second metric is the average anomaly rank (**rank**). It was introduced in Section 2.2.4 and indicates the anomaly detector's ability to assign the anomalous examples a high anomaly score, relative to the others. A perfect detector has a rank of 1. The lower the rank, the better the predictions.

5.5 Visualisation

The last step of the pipeline is an interactive explorer which is available online: <http://cern.ch/anomaly-detection-mki>. It allows to inspect both the input data and predictions with live metrics. A screenshot is shown in Figure 5.7.

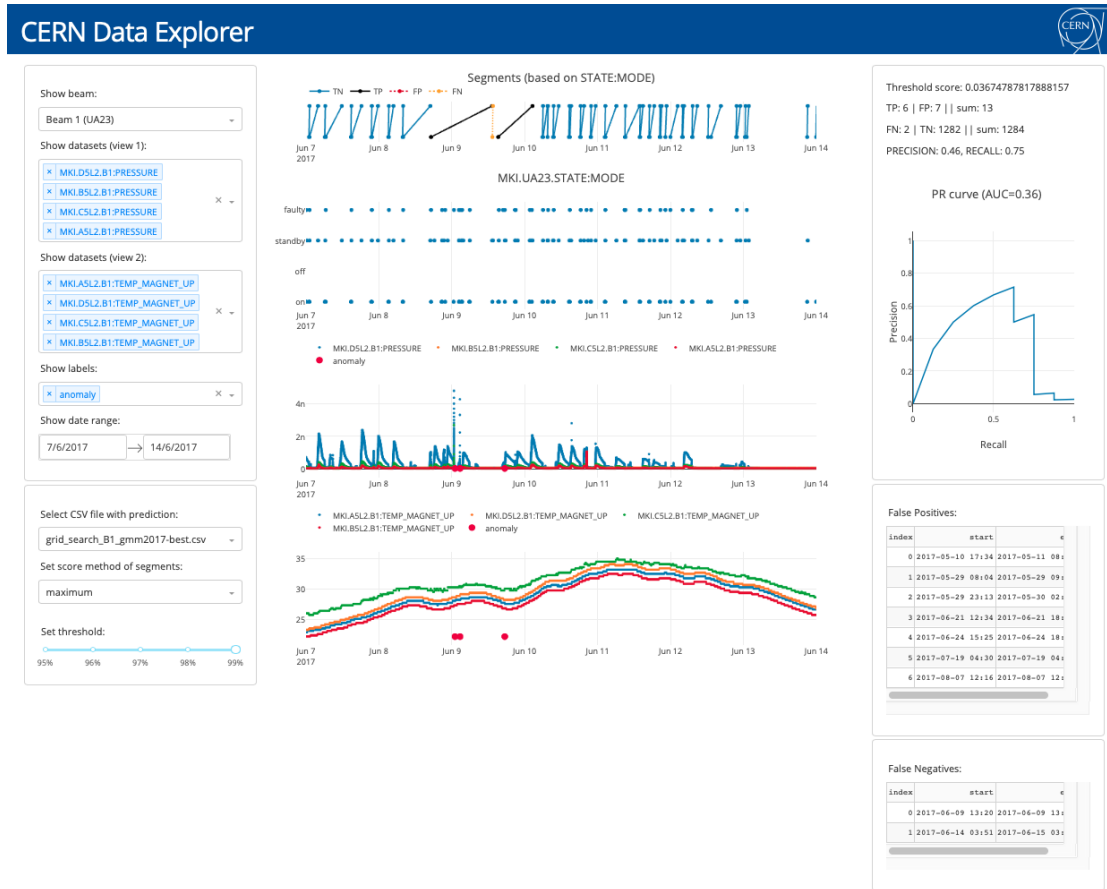


Figure 5.7: Visualisation of the predicted segments. From June 7 until June 14 (2017), there are two TPs and one FN. The other segments are TNs as is the majority of the segments. The anomalies are shown together with the **PRESSURE** and **TEMP_MAGNET_UP** data. There are spikes in the **PRESSURE** data close to the first two anomalies.

5.5.1 Data exploration

The explorer offers the functionality to inspect all datasets together with the e-logbook. Since it is interactive, a user no longer has to generate plots manually for certain data sets. Because of the large amount of variables, this was a tedious process before. It is also easy to adapt the date range.

An additional bonus is that a certain configuration of shown datasets for a selected date range and labels can easily be shared via a URL since the state is stored explicitly as a parameter. Sharing this link with someone else makes it unnecessary to share screenshots when discussing certain data.

5.5.2 Predictions

Besides the input data, the explorer also shows the segments with their predictions. They are given a color corresponding to the category of the confusion matrix (see Section 2.2.4).

A user can easily change certain parameters. For example the segment score method and decision threshold can be adapted. The anomaly detector itself can be changed by choosing another prediction file.

The effect of these changes will automatically be shown in the explorer. The predicted segments, confusion matrices and PR curve will be updated.

Chapter 6

Experiments and results

The implementation of the anomaly detection application is written in Python. It uses Scikit-learn [16] for the anomaly detectors and Dash by Plotly [9] for the visualisation. This chapter discusses the results of four types of experiments.

First, an optimal set of hyper-parameters is determined in a *grid search*. It leads to a configuration of the detector which will likely yield good results on future predictions.

Second, a *feature selection* algorithm is performed to reduce the number of features used for training and predicting. This is useful because it might yield a subset of features giving better performance with a lower computational intensity for training the model.

Third, we move from predictions in an offline to an *online setting*. Instead of having all historical data, the amount of data is increased week per week. A model is retrained at the end of the week and is then used to make predictions for the following week. This is useful since it is an important first step towards the integration of the application in the control room at CERN.

Finally, a method to *incorporate expert feedback* into the anomaly detection application is proposed and evaluated. The idea is that an expert could interactively cluster the output segments of the predictions of historical data, and re-use the knowledge of this clustering for future predictions. If this knowledge can be transferred from one year to another, it could be useful for reducing the number of false positives which would increase the precision of the predictions.

Scripts for reproduction, together with the code and CERN data are available in the GitLab repository¹. Under the directory `src/scripts/text_results` the log files and detailed information of the results can be found.

¹<https://gitlab.cern.ch/te-abt-ec/anomaly-detection-mki-2019>

6.1 Global decisions

6.1.1 Anomaly detectors

In the experiments, we compare the predictions of the applications using two models: GMM and iFOREST. The model used is determined by the parameter `anomaly_detector`. The corresponding hyper-parameters can be specified in `detector_params`.

6.1.2 Thresholds

When using an anomaly detector, one has to decide on the threshold (`threshold`) for predicting a segment as anomalous or not. In the developed framework it can be set relatively to the scores generated by the model. Although it might be beneficial to adjust the threshold according to the data in the specific time interval, in practice a certain threshold must be chosen.

For all experiments, `threshold` is set to 99% for GMM and 95% for iFOREST. The threshold for GMM is higher, because the detector in general performed better than iFOREST. Setting a higher threshold for iFOREST would have resulted in a lower recall, whereas setting a lower threshold for GMM would have resulted in a lower precision.

6.1.3 Segment score

The segment score method is set to `max`. Compared to `top_k` and `top_percentage` it is a more “aggressive” setting since it flags a segment as anomalous if it contains a single instance with a high anomaly score. Although this decision might lead to a higher number of FPs, it is expected to have a lower number of FNs. This is reasonable because it is more important for CERN not to miss an anomaly than to have a possibly false alarm.

6.1.4 Beam

To make the experiments manageable, the data used in the experiments is limited to beam 1. Beam 1 is preferred over beam 2 because it has more anomalies (see Table 3.9). This is beneficial because making statements based on the evaluation of more anomalies is stronger.

6.2 Grid search

The performance of a machine learning model is dependent on the choice of its hyper-parameters (see Section 5.2.1). To automate the process of making the right decision, a grid search running through several combinations was developed. In this experiment, we derive the most performant hyper-parameter choice for each anomaly detector per year. Next, we propose a general set of hyper-parameters for future use.

An example call to the grid search method looks as follows:

Listing 6.1: Grid search

```

1 results = grid_search(
      feature_selection=feature_selection,
3      beam=beam,
      scale_data=scale_data,
5      anomaly_detector=anomaly_detector,
      detector_params=detector_params,
7      labels=labels,
      segment_score=segment_score,
9      x_train=df,
      state_mode=state_mode,
11     scoring=scoring
    )

```

We explain each of the arguments:

- **feature_selection**: the subset of columns of `x_train` used for training and prediction
- **beam**: the beam for which the grid search is executed
- **scale_data**: the scaling method to use during preprocessing
- **detector_params**: the configuration of all hyper-parameters to evaluate; formatted as a dictionary where values are of type list
- **labels**: the e-logbook labels to use during evaluation
- **segment_score**: the segment score method
- **x_train**: the training instances
- **state_mode**: the state mode data for the test instances
- **scoring**: the metric to use for the evaluation; either AUC or rank

6.2.1 Configuration

A set of hyper-parameters (`detector_params`) is tested while keeping the other parameters constant. All features of the instances are used for training and prediction, so there is no feature selection involved (this will be done in Section 6.3). The AUC metric is used because it is standard in the field of anomaly detection.

GMM

For GMM the following parameters are tested:

```

n_components = [1, 2, 4, 6, 8]
2 covariance_type = ['full', 'tied', 'diag']
n_init = [1, 5]
4 init_params = ['kmeans']
scale_data = ['standard']

```

Combining these parameters results in 30 configurations.

We have limited the number of `n_components` to limit the training time and to avoid overfitting. The larger the number of `n_components`, the more specific relations in the data will be learnt but the less it will generalize. We assume that 8 clusters are sufficient.

For `covariance_type` we excluded `spherical` because it did not perform well. This has also been suggested in previous work [8].

The number of initialisations is limited to reduce the execution time of the grid search. An option `n_init=5` is added for verification.

For `init_params`, we have excluded `random`. Although `random` sometimes led to a good AUC, we were not able to reproduce good results with the same configuration.

For `scale_data`, we have excluded `none` because GMM depends on scaling. In small set-ups we observed that in fact not using any scaling yielded worse results.

iForest

For iFOREST the following parameters are tested:

```
1 n_estimators = [50, 100, 200]
  max_samples = [256, 256*10, 256*20]
3 contamination = [0.005, 0.01]
  max_features = [1.0]
5 scale_data = ['standard', 'none']
```

Combining these parameters results in 36 configurations.

For `n_estimators` the default value is 100. We have halved and doubled this value.

For `max_samples` the default value is 256. Halilovic has suggested [8] that larger values perform better. Although the performance of iFOREST was not satisfying, we have also included larger values.

For `contamination` the default value is 0.01. However for our problem, the contamination is lower. On the level of segments for 2017, there were 8 anomalous segments and 1297 normal segments. This is a contamination of 0.6%. On the level of individual instances this will be even lower, because not always all instances of a segment render it anomalous. Therefore we included a lower value for the contamination.

For `max_features` we did not allow a subset of features, since we will consider this effect later during feature selection (see Section 6.3).

For `scale_data` we allow both scaling and no scaling. This is because iFOREST in general does not depend on scaling.

6.2.2 Results

GMM

Per year For every year, we executed a grid search. This grid search executes the pipeline for all combinations of parameters. Each of these executions is evaluated

Table 6.1: Optimal parameters for GMM per year as result of a grid search with beam=1, segment_score=max.

Parameter	2016	2017	2018
n_components	4	8	2
covariance_type	full	full	full
n_init	5	5	5
init_params	kmeans	kmeans	kmeans
scale_data	standard	standard	standard

Table 6.2: A general set of parameters for GMM.

Parameter	Value
n_components	2
covariance_type	full
n_init	1
init_params	kmeans
scale_data	standard

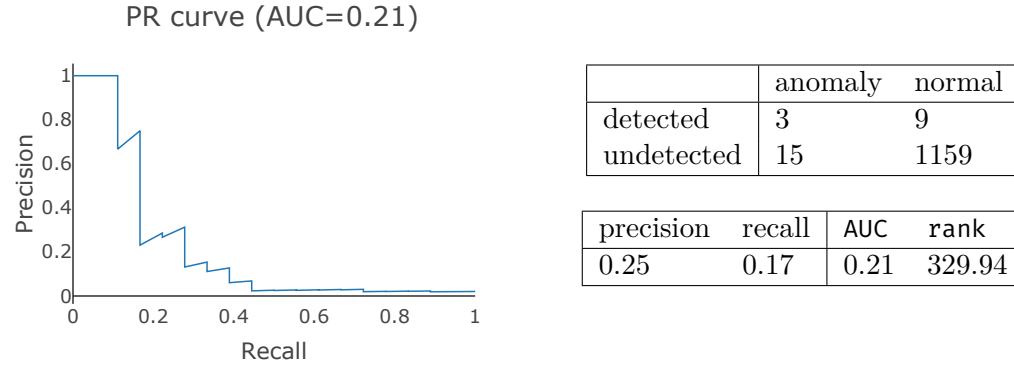
according to its AUC. The best result of the grid search per year (i.e. the configuration with the best AUC) is shown in Table 6.1. We will refer to these configurations as the best set of parameters per year. The corresponding PR curves and predictions are shown in Figure 6.1.

For both 2017 and 2018 the model detects respectively 63% and 57% of the anomalies. The good values for AUC and rank confirm that the model is actually able to detect the anomalies.

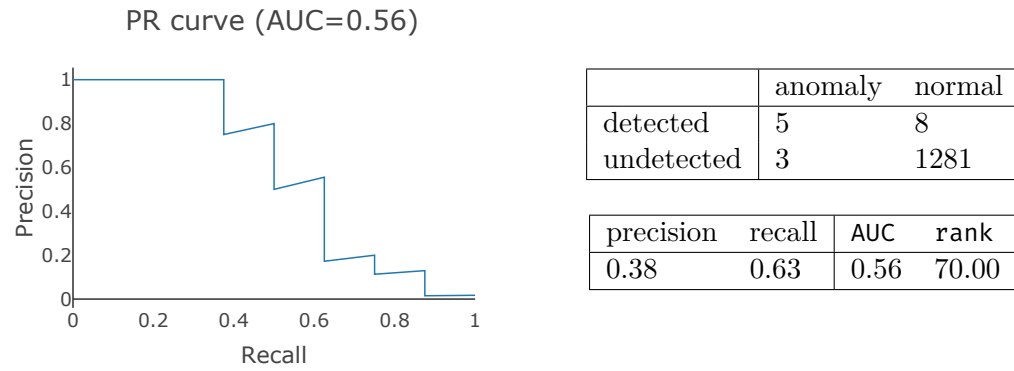
For 2016 however only 17% of the anomalies are detected. The AUC and rank metric are also considerably worse than for 2017 and 2018.

Generalisation Although we have found an optimal set of parameters per year, we require also a general set of parameters which will likely yield good results. We suggest the configuration in Table 6.2 for this purpose. We designed this configuration not to use many components. This should lead to learning behaviour and not to overfitting. It is also substantiated by a theoretical derivation of the optimal number of components by Wéry [21]. Further, it will be faster for training and predicting which is useful for the next experiment. For the other settings, we inspected the results of the grid search. The boxplots in Figure 6.2 turned out to be useful. Similar boxplots with rank as metric (not shown) were also used.

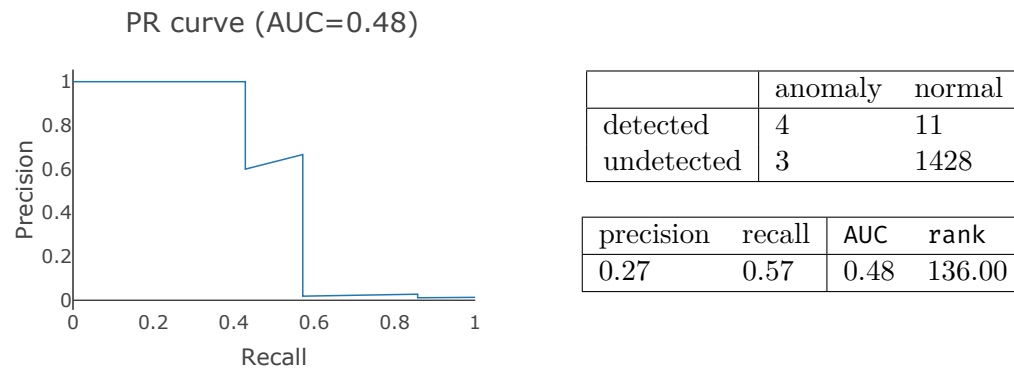
6. EXPERIMENTS AND RESULTS



(a) 2016



(b) 2017



(c) 2018

Figure 6.1: PR curves and predictions of the best results after a grid search with `anomaly_detector=GMM`, `beam=1`, `segment_score=max`, `threshold=99%`.

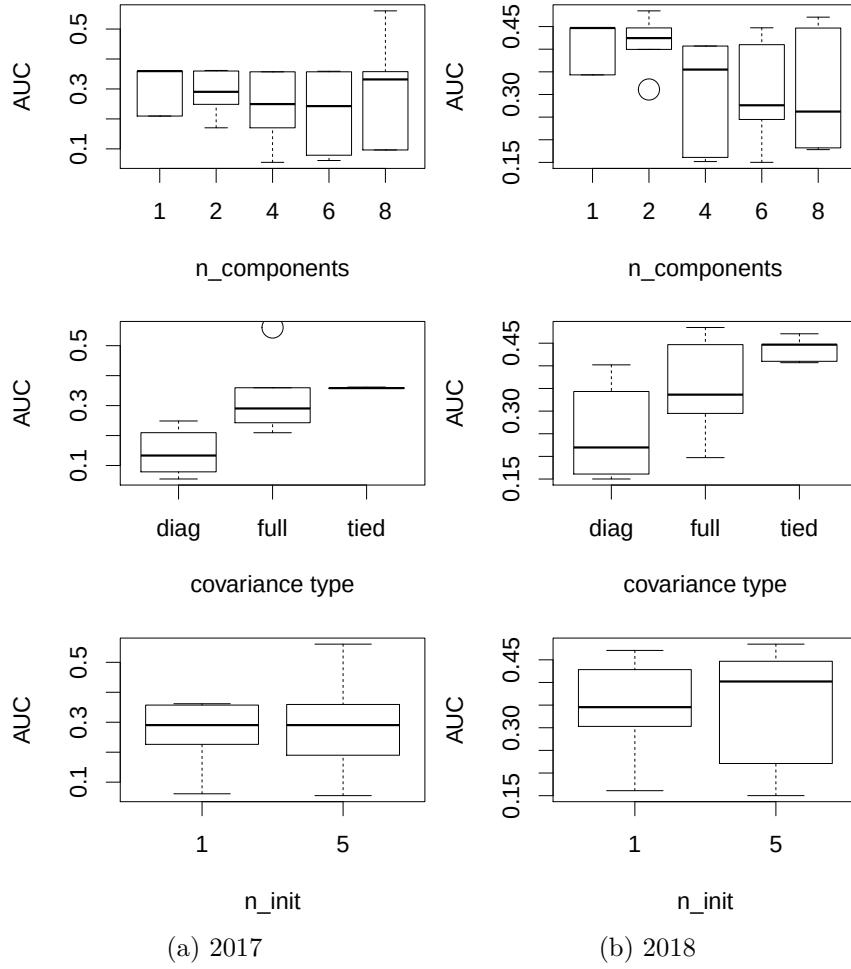


Figure 6.2: Grid search results for 2017 and 2018, with settings `anomaly_detector=GMM`, `beam=1`, `segment_score=max`.

iForest

Per year The grid search with iFOREST resulted in the optimal parameters per year as shown in Table 6.3. The corresponding confusion matrices and predictions are shown in Figure 6.3.

For 2016, 2017 and 2018 respectively 28%, 38% and 29% of the anomalies are detected. However, the corresponding precision is lower than for GMM. The AUC and rank are also worse than for GMM. Although iFOREST correctly detects some anomalies, it is outperformed by GMM.

Generalisation The results obtained with iFOREST are not satisfactory. For completeness, we show the spreading of the data for the executed grid search in Figure 6.4. Although there are some slight differences, for the first three parameters we argue that these do not bring any value. However the `scale_data` plot suggests

Table 6.3: Optimal parameters for iFOREST per year as result of a grid search with beam=1, segment_score=max.

Parameter	2016	2017	2018
n_estimators	50	100	200
max_samples	256	5120	256
contamination	0.005	0.01	0.005
scale_data	standard	standard	standard

Table 6.4: A general set of parameters for iFOREST.

Parameter	Value
n_estimators	100
max_samples	256
contamination	0.01
scale_data	standard

that using scaling is beneficial. We conclude to use scaling, but it remains to be seen whether a set of hyperparameters can actually give us good performance. We propose the general set of parameters in Table 6.4.

6.2.3 Conclusion

The grid search experiment has yielded an optimal set of parameters per year and a general set of parameters for both GMM and iFOREST. By comparing the predictions with both detectors, we have concluded that GMM outperforms iFOREST.

6.3 Feature selection

In a second experiment, we apply feature selection to reduce the number of features used for training and predicting. This is useful because it might yield a subset of features resulting in better performance and a lower computational intensity.

6.3.1 Set-up

Feature selection [7] is the process of selecting a subset of features to use for the training of a machine learning model. A first idea would be to try all subsets in a brute-force search. However already without the generated features, our dataset consists of 120 variables. Since a set of n elements has 2^n subsets, this is not feasible. Therefore we propose a greedy local search algorithm.

Greedy local search

The feature selection starts with an initial solution. This might be an empty set of features or a fixed set of features. Using this solution a set of neighbours is

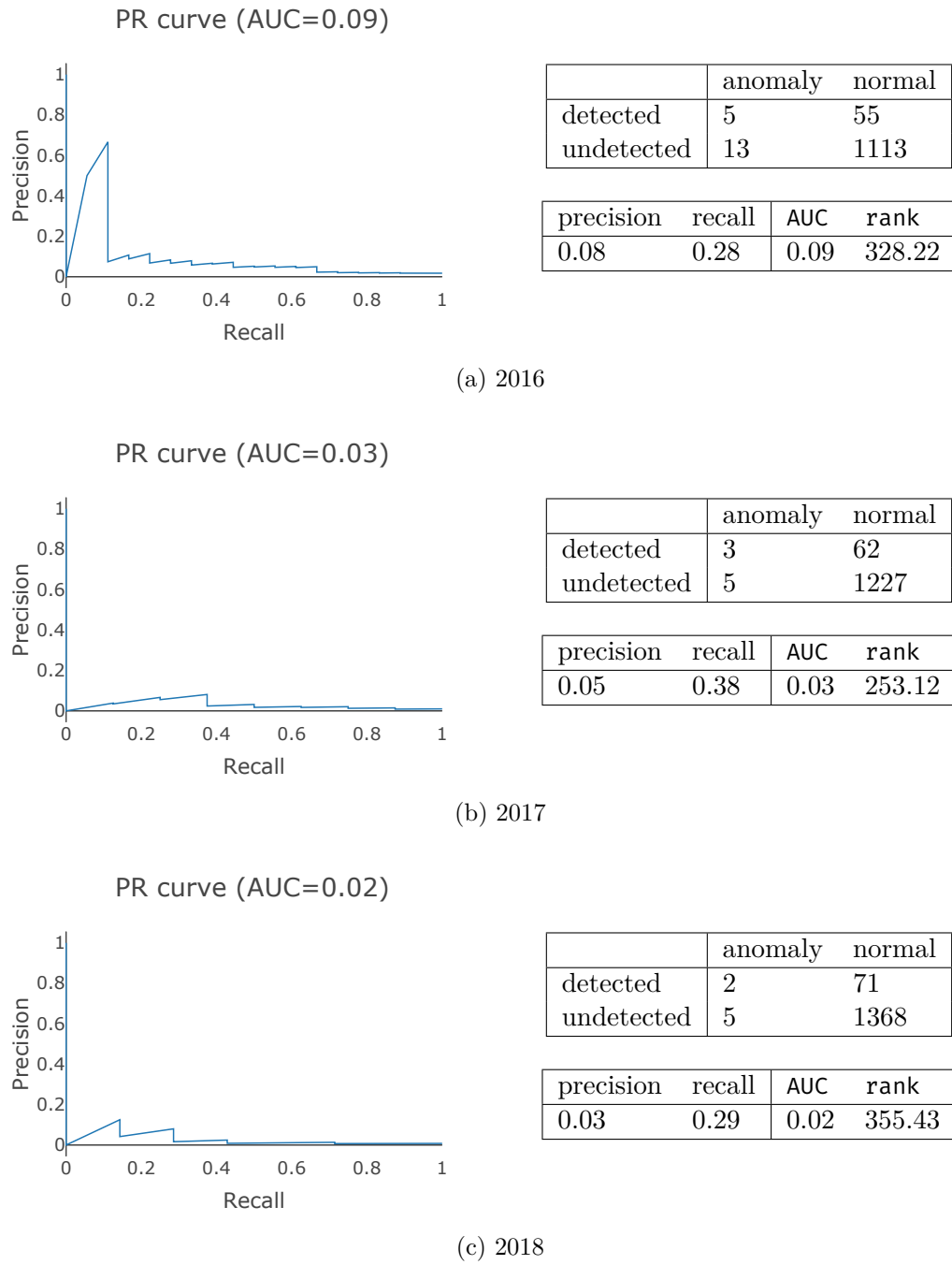


Figure 6.3: PR curves and predictions of the best results after a grid search with `anomaly_detector=IFOREST`, `beam=1`, `segment_score=max`, `threshold=95%`.

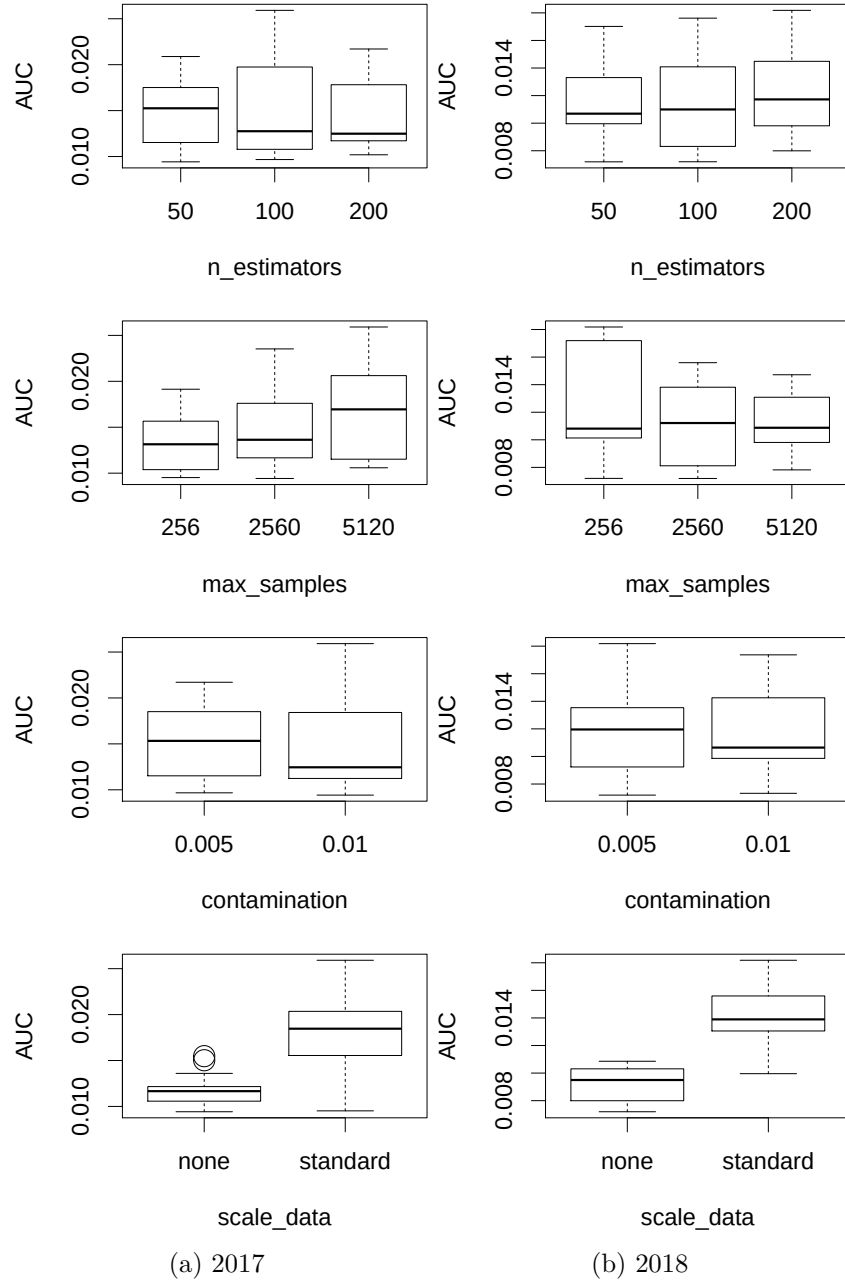


Figure 6.4: Grid search results for 2017 and 2018, with settings `anomaly_detector=iFOREST`, `beam=1`, `segment_score=max`.

determined. We define two types of neighbours:

- Type 1: remove a feature from the current solution.
- Type 2: add a feature to the current solution (which is not in the current solution).

For all neighbours a cost will be computed. The cost function can be configured as the **rank** or **inverse AUC**, resulting from a **GMM** or **iFOREST** pipeline.

The neighbour with the lowest cost (lowest **rank** or highest **AUC**) will be selected as the next current solution. This process iterates until a maximum number of iterations has been executed. Finally the best solution of the last iteration will be returned.

Listing 6.2: Feature selection

```

1 for i in range(0, max_iterations):
    neighbors = get_neighboring_solutions(solution, use_features)
3    solution, df, score = select_neighbor(neighbors, compute_score)

```

Initial solution

As initial solution, we selected a set of 10 variables. We decided to choose these variables, since they have shown to be a good decision in the first runs of the algorithm (starting from no variables). Thanks to this initial solution, the algorithm will quicker find a solution. However, one should be cautious that we might have led the algorithm into a local optimum. One could decrease the chance of this problem by introducing other neighbour types which differ in for example more than one feature. We decided not to do this because of computational restrictions.

The initial solution is:

```

1 'MKI.C5L2.B1:TEMP_MAGNET_UP_fft_freq_max',
  'MKI.A5L2.B1:TEMP_TUBE_DOWN_fft_freq_max',
3 'MKI.UA23.IPOC.CB1:T_DELAY',
  'LHC.BCTFR.A6R4.B1:BEAM_INTENSITY',
5 'MKI.UA23.STATE:SOFTSTARTSTATE',
  'MKI.UA23.F3.CONTROLLER:KICK_LENGTH_TOPLAY',
7 'MKI.UA23.STATE:CONTROL',
  'MKI.B5L2.B1:PRESSURE:SW_MEAN_DIFF_600_s',
9 'MKI.B5L2.B1:TEMP_TUBE_UP',
  'MKI.B5L2.B1:TEMP_TUBE_UP:SW_MEAN_DIFF_600_s'

```

Parameters

Since some optimal configurations require a substantial amount of computation time (e.g. `n_components=8`), the parameters are configured according to the general set of parameters. For **GMM** and **iFOREST** it is shown in respectively Table 6.2 and 6.4.

Table 6.5: Results of the feature selection experiment. The progress of the algorithm is shown in Figure 6.6.

(a) GMM

	2016	2017	2018
Number of features	12	12	16
rank	51.00	9.25	13.57
Relative rank to Fig. 6.1	15.46%	13.21%	9.98%
PR curves and predictions	Fig. 6.5a	Fig. 6.5b	Fig. 6.5c

(b) iFOREST

	2016	2017	2018
Number of features	16	26	24
rank	81.06	43.00	81.86
Relative rank to Fig. 6.3	24.70%	16.99%	23.03%
PR curves and predictions	Fig. 6.7a	Fig. 6.7b	Fig. 6.7c

Stopping criterion

The stopping criterion is the number of iterations. Since each step of the feature selection evaluates many neighbours, an iteration is expensive (in execution time). The number of iterations is fixed at 20 which is a priori an arbitrary decision. A posteriori the results will suggest that this amount is sufficient.

6.3.2 Results

The resulting cardinalities of the feature subsets with their corresponding **rank** are listed in Table 6.5. The optimal feature subsets themselves are given in Appendix B. The resulting predictions can be compared to the predictions with optimal hyper-parameters per year on all features (Figure 6.1 and 6.3).

We see that by using at most 16 features for GMM and 26 features for iFOREST the **rank** has decreased. For example for GMM in 2017 the rank has decreased from 70.00 to 9.25. This is 13.21% of the original rank. We have added these reductions for all results as the relative **rank**. In all set-ups, we observe a significant reduction. It means that using a subset of features, helps the detector to assign higher anomaly scores to the more anomalous instances.

GMM which already performed better after the grid search, also gives the best absolute and relative **rank** values. When inspecting the predictions, we now see that GMM detects 88% and 71% of the anomalies in 2017 and 2018.

Additionally we see that for both detectors over all years, the recall values are identical or better (i.e. at least the same amount of anomalies could be discovered as with all features).

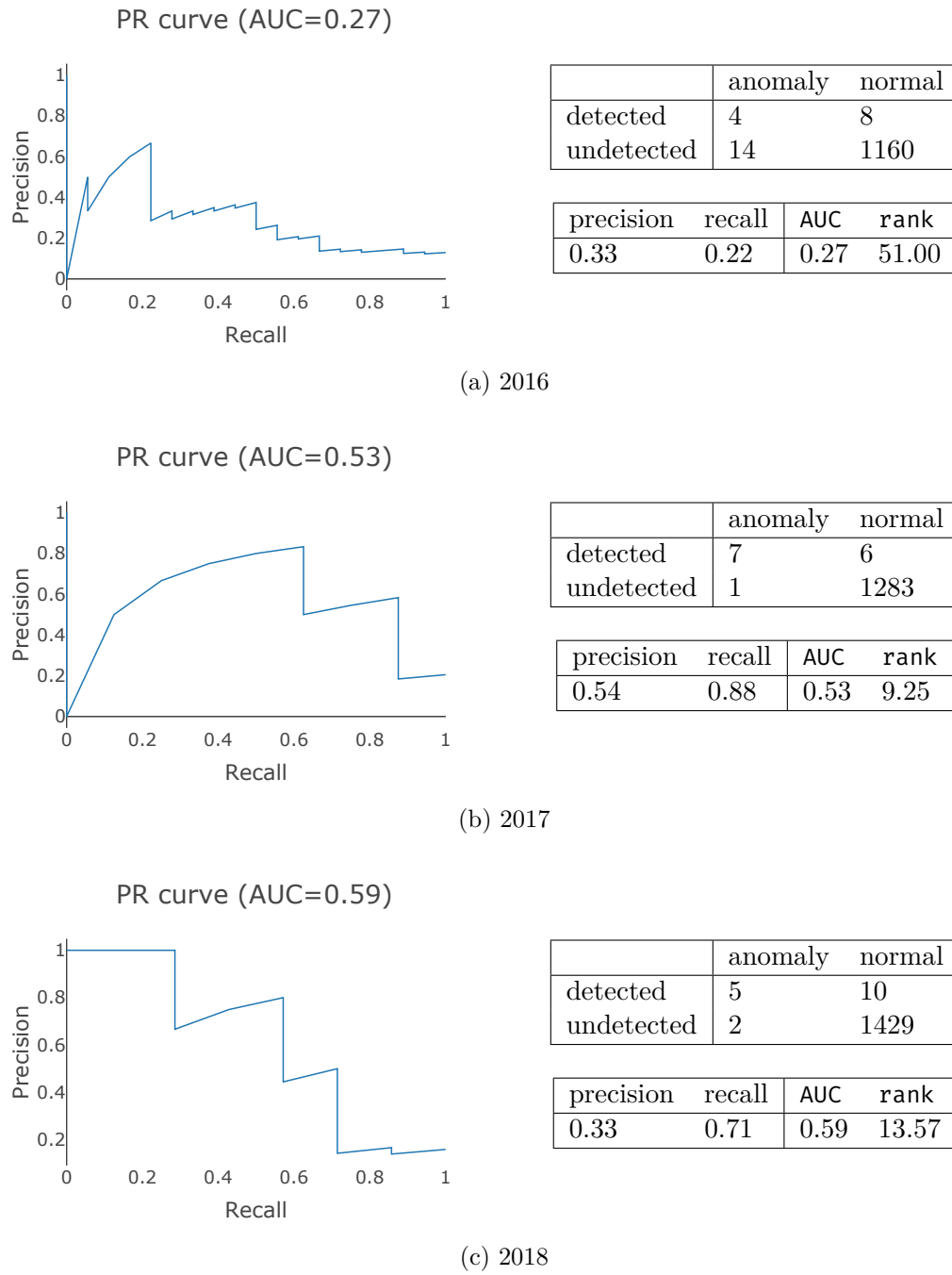
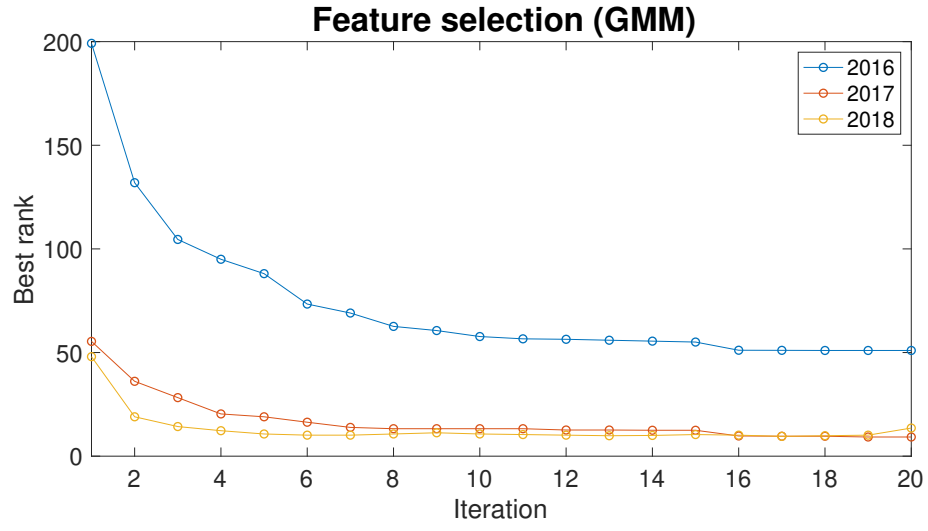
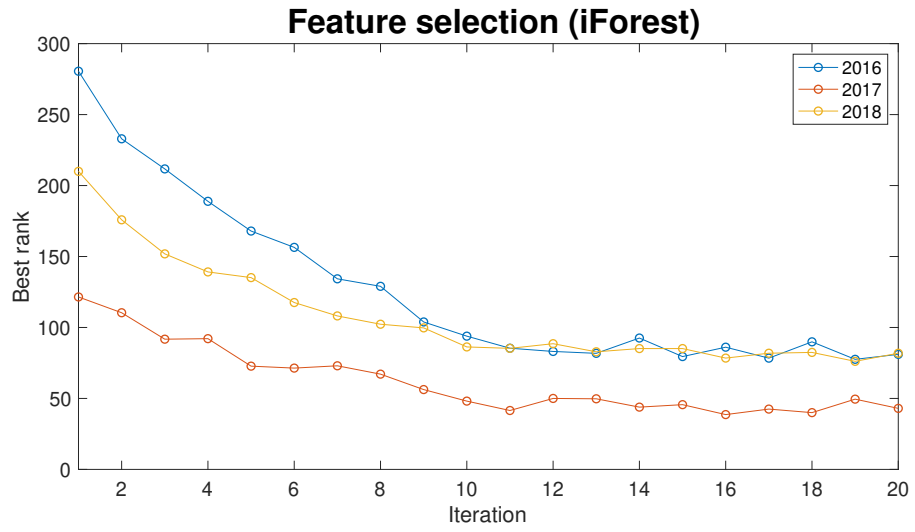


Figure 6.5: PR curves and predictions of the best results after feature selection with `anomaly_detector=GMM`, `beam=1`, `segment_score=max`, `threshold=99%`.

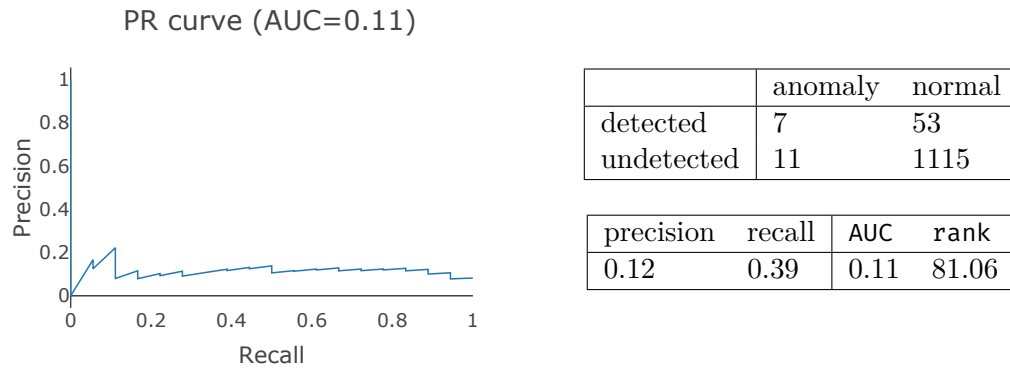


(a) GMM

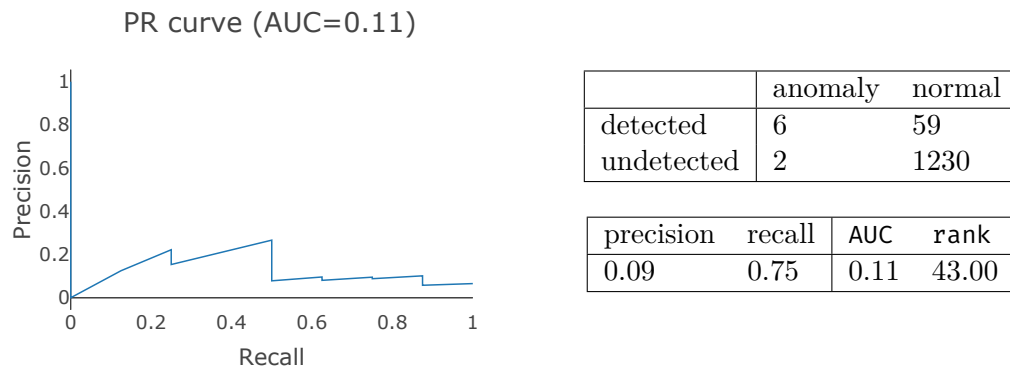


(b) iFOREST

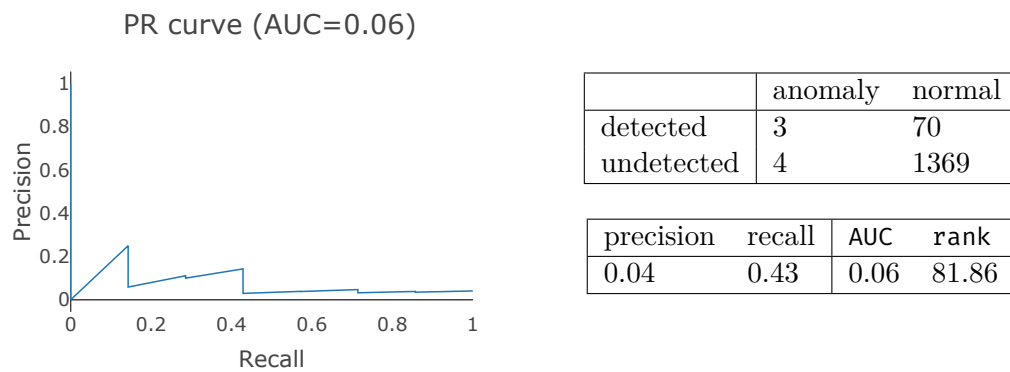
Figure 6.6: The progress of feature selection for both anomaly detectors on the data of beam 1.



(a) 2016



(b) 2017



(c) 2018

Figure 6.7: PR curves and predictions of the best results after feature selection with `anomaly_detector=IFOREST`, `beam=1`, `segment_score=max`, `threshold=95%`.

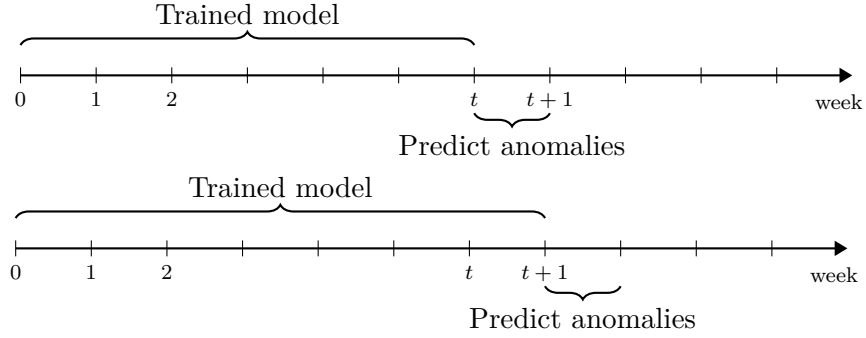


Figure 6.8: Live evaluation: set-up. For predictions during week t a model trained on historical data from week 0 until the last day of week $t - 1$ is used. At the beginning of week $t + 1$ the model is retrained incorporating the additional data of week t .

6.3.3 Conclusion

For both anomaly detectors, feature selection helps to increase the quality of predictions. It suggests that a subset of features is sufficient. This is beneficial for the computational power needed to train the model. As for the grid search, the GMM model is still preferred.

6.4 Live evaluation

In a third experiment, we aim to make the circumstances for the anomaly detection application as realistic as possible. We move from predictions on historical data to an online setting. Therefore this experiment is called live evaluation. The knowledge for training a model for future predictions is gradually increased as time goes on.

This experiment is useful since it is closer to the actual usage of the anomaly detection application in the control room at CERN. It is the first time that an experiment reflecting the real-world scenario has been performed.

6.4.1 Set-up

Concretely, we have discretized the live evaluation into weeks. The experiment starts in week 1, when CERN is given a model that was trained based on historical data of week 0. It is used to predict anomalies during this week. At the end of week 1, CERN trains a new model incorporating the additional data. This model is then used for prediction during the next week. This process is iterated every week. A graphical representation is shown in Figure 6.8.

Notice the difference with the offline setting where a model is trained on historical data and then used to make predictions on this data. In the live evaluation setting, it is harder to train a model since it only accesses a local part of the data in the beginning. The more time progresses, the better we assume the model will perform since it is trained on a larger set of data.

Table 6.6: Live evaluation: set-up for 2017 and 2018.

	First segment	Last segment	Number of prediction weeks
2017	April 12	December 4	33
2018	March 23	December 3	36

Anomaly detector

Since performing live predictions is a more challenging task, we decided to do this experiment for the best model only. Based on the previous two experiments, we continue with GMM.

Features

The experiment was performed for two feature sets. The first one contains all features, while the second one contains only the optimal feature subset (derived in Section 6.3).

Hyper-parameters

To capture precisely the effect of moving from an offline to an online setting, it is necessary to compare the online results with a similar offline setting. In previous (offline) experiments, we have obtained results with both a grid search and feature selection. Since these experiments used different sets of parameters, we apply the same distinction to this experiment:

- For all features, we use the parameters derived in the grid search experiment (Table 6.1). This allows to directly compare the results to Figure 6.1.
- For the optimal feature subset, we use the general parameters (Table 6.2). This allows to directly compare the results to Figure 6.5.

Weeks

In the postprocessing step segments are created based on the `STATE:MODE` data. Table 6.6 shows the date of the first and last segment for both 2017 and 2018. Week 0 begins at the day of the first segment. As explained above, it is used to train the first model which is used for predictions during the consecutive week.

For later interpretation of the results it is important to know that there are no anomalies during week 0 for both 2017 and 2018. This implies that all anomalies can still be predicted.

Notice that the number of segments considered and predicted will be slightly different from the historical data. First, the segments of week 0 will not be predicted and are therefore left away. We did this because in week 0 there is no data yet, and thus no model. Alternatively one could use a model based on the historical data of the previous year. We decided not to do this because the data of different years is quite different due to interventions during e.g. the YETS (see Section 2.1.4). Second, segments accidentally overlapping between two weeks will influence the total number of segments.

Table 6.7: Confusion matrices and metrics of the results after live evaluation on *all features* with `anomaly_detector=GMM`, `beam=1`, `segment_score=max`, `threshold=99%`. Hyperparameters are set per year to the best results of the grid search (see Table 6.1).

(a) 2017			(b) 2018		
	anomaly	normal		anomaly	normal
detected	2	31	detected	4	32
undetected	6	1253	undetected	3	1376

precision	recall
0.06	0.25

precision	recall
0.11	0.57

Table 6.8: Confusion matrices and metrics of the results after live evaluation on *all features* with `anomaly_detector=GMM`, `beam=1`, `segment_score=max`, `threshold=95%`. Hyperparameters are set per year to the best results of the grid search (see Table 6.1).

(a) 2017			(b) 2018		
	anomaly	normal		anomaly	normal
detected	5	74	detected	4	85
undetected	3	1210	undetected	3	1323

precision	recall
0.06	0.63

precision	recall
0.04	0.57

6.4.2 Results

All features

The aggregated results of the live evaluation using GMM with all features are shown in Table 6.7. We compare these results with the offline setting of Figure 6.1.

For 2017, the online application does not succeed to detect as many anomalies as offline. Two instead of five anomalies are detected, leading to a decrease of recall from 0.63 to 0.25. As a sanity check, we have decreased the `threshold` for GMM to 95% to see how this would impact the result. As hoped, we see in Table 6.8 that the recall increases again to 0.63 and that the precision remains the same. It is an arbitrary choice to change to `threshold` but possibly 99% is too strict for a live setting.

For 2018, the live evaluation detects the same amount of anomalies. The recall remains 0.57 but precision decreases from 0.27 to 0.11.

Table 6.9: Confusion matrices and metrics of the results after live evaluation on the optimal *feature subset* (derived in Section 6.3) with `anomaly_detector=GMM`, `beam=1`, `segment_score=max`, `threshold=99%`.

(a) 2017			(b) 2018		
	anomaly	normal		anomaly	normal
detected	1	32	detected	3	33
undetected	7	1252	undetected	4	1375

precision	recall
0.03	0.13

precision	recall
0.08	0.43

Table 6.10: Confusion matrices and metrics of the results after live evaluation on the optimal *feature subset* (derived in Section 6.3) with `anomaly_detector=GMM`, `beam=1`, `segment_score=max`, `threshold=95%`.

(a) 2017			(b) 2018		
	anomaly	normal		anomaly	normal
detected	4	75	detected	4	85
undetected	4	1209	undetected	3	1323

precision	recall
0.05	0.50

precision	recall
0.04	0.57

Subset of features

The aggregated results of the live evaluation using GMM with the optimal subset of features are shown in Table 6.9. We compare these results with the offline setting of Figure 6.5.

For 2017, the online application detects only 1 instead of 7 anomalies. This is a decrease of the recall of 0.88 to 0.13. Again, we performed the experiment with a lower threshold of 95% (see Table 6.10). It detects 3 additional anomalies, but still fails to detect 3 anomalies which were found in the offline mode.

For 2018, the online application detects 3 anomalies, compared to the 5 anomalies in offline mode. However with the decreased threshold, 1 more anomaly is detected.

6.4.3 Conclusion

We have subjected the anomaly detection application to an online prediction setting. As expected, results are slightly worse than for an offline setting but by decreasing the original threshold it is still able to detect most of the anomalies as detected offline. However, one must carefully select the threshold and cope with the resulting FPs.

6.5 Incorporating expert feedback

Most results of previous experiments have a high recall. This means that most anomalies were actually detected. However, sometimes the precision might be low which means that there are quite some false alarms.

Although it is more important for CERN not to miss anomalies (FN) than to give a false alarm (FP), a low precision is not desirable. FPs are situations where an alarm is caused, while there is no anomaly. It is clear that the model somehow fails. It cannot classify a normal situation as not anomalous.

To solve this problem, we propose an additional step that interactively clusters the output of the anomaly detector using COBRAS (see Section 2.4). We aim to correct mistakes of the model by re-assigning it to the correct cluster of anomalous or normal segments. This knowledge is stored and re-used for later predictions. We have two reasons to introduce this step instead of adjusting the existing model.

First, it is hard for humans without expert knowledge to understand why certain segments are anomalous and others are not. There is a need for expert knowledge to actually classify these anomalies as normal situations. It concerns segments where we expect a good model to actually predict these cases as anomalies.

Second, we want to incorporate expert knowledge independent of the detector itself. We do not want to give up the flexibility of using any anomaly detector.

So, in this fourth experiment we explain how the proposed clustering method works and discuss the results. It is useful because including expert feedback appears to be necessary. Our approach to include expert feedback is new.

6.5.1 Interactive clustering

Initially the anomaly detection application is used to predict segments as anomalous or not. It is likely that there will be some FPs after comparing the predictions with the e-logbook. We aim to cluster the predicted segments using COBRAS.

COBRAS segments The available COBRAS implementation² requires a fixed length representation per segment which is assumed to be useful for clustering. Since the segments of the predictions, consist of data for a varying number of instances some choice must be made. We decided to create so-called COBRAS segments based on the prediction segments. It is important to carefully select their attributes. We observed that for example the frequency of the pressure is an important indicator of an anomaly. A large difference between the minimum and maximum value often predicts an anomaly (see for example Figure 6.12b). Therefore we included this type of aggregated attributes. All attributes are shown in Table 6.11.

Labels Besides the data attributes, each COBRAS segments has a label. The label is the predicted class for a fixed threshold. It can be TP, FP, FN or TN.

²https://bitbucket.org/toon_vc/cobras_ts

Table 6.11: Attributes of a segment in COBRAS.

Attribute	Explanation
nb	The number of instances in the segment
y_pred	The anomaly score of the segment
freq_temp	The maximal frequency of the temperature
min_freq_pressure	The minimal frequency of the pressure
max_freq_pressure	The maximal frequency of the pressure
timestamp_min	The start of the segment
timestamp_max	The end of the segment

629		949	
cluster_pred	2	cluster_pred	1
nb	145	nb	37
y_pred	1	y_pred	0.00682545
freq_temp	1.3377	freq_temp	-0.20094
min_freq_pressure	1.1524	min_freq_pressure	-0.396986
max_freq_pressure	1.1524	max_freq_pressure	-0.396986
timestamp_min	2017-08-07 12:16:54.495000124	timestamp_min	2017-09-25 15:25:10.180000067
timestamp_max	2017-08-07 12:41:01.894000053	timestamp_max	2017-09-25 18:17:03.953999996
url	Open in explorer	url	Open in explorer
y_true	0	y_true	0
Should segment 629 (FP) and segment 949 (TN) be in the same cluster? [Y/n]			

Figure 6.9: The system queries the user: “*Should segment X and segment Y be in the same cluster?*”. Automatically generated links guide the expert to the segments in the explorer, where pressure and temperature datasets are pre-selected.

Questions During the clustering, COBRAS asks questions of the format: “*Should segment X and segment Y be in the same cluster?*” (see Figure 6.9). A hybrid querier is added around COBRAS to assume that only FPs might possibly need to be re-assigned to another cluster. It is called hybrid because it answers questions concerning non-FP segments automatically with *no*, but requires interaction for questions concerning at least one FP segment. The user should answer yes to a query if the given FP is actually a false alarm and is proposed to be clustered with a FN. To decide this, the data of the segment must be inspected by an expert. An example is shown in Figure 6.10 where an actual FP is compared to a FN. The expert should cluster these segments together, to reduce the number of FPs.

Although in the nominal case we expect the comparison of a true FP with a FN, there can be queries about any combination with a FP. We discuss two special cases.

6. EXPERIMENTS AND RESULTS



(a) Segment 916 (FP)



(b) Segment 905 (TN)

Figure 6.10: The FP does not show any spikes in the data. The data pattern is similar to the given TN. The expert should answer the query with *yes*, with the aim to cluster the FP as a TN.

Special case 1 In Figure 6.11 a FP (with clear spikes in the pressure data) is compared to a TP. Although there was no event in the e-logbook, CERN experts have indicated that these events are also anomalous and therefore of interest. The expert should cluster these segments together with TPs, to ensure that these events trigger an alarm.

Special case 2 In Figure 6.12 a FP is compared to a TN. In the normal case, we would like to assign the FP to the cluster of the TNs. However, in this case the FP contains a spike and is anomalous, although there is no log. This case must trigger an alarm and therefore the expert should answer with no.

Store the information When the query budget of COBRAS is exhausted, it will present a clustering. This clustering, together with the set of input segments is stored for later use.

6.5.2 Re-use of the clustering

Now we assume to have predictions for segments of another time interval, obtained with the pipeline of Chapter 5 (without clustering). The goal is to improve these predictions with the additional knowledge incorporated in the COBRAS clustering. To do so, we first train a k-nearest neighbour classifier with the previous COBRAS clustering. Then we assign every new segment to its nearest neighbour. If this is a TP or FN, the predicted anomaly score is set to 1. If it is a FP or TN, the predicted anomaly score is set to 0. Now these predicted segments are the improved predictions. They can be evaluated and compared to the previous predictions without COBRAS improvement.

6.5.3 Experiment

Set-up

As an experiment we have predicted the 2017 segments using the pipeline. Then, we interactively clustered this predicted segments using COBRAS with a query budget of 250. It means that COBRAS obtained knowledge of the relation between 250 segments. Most of these queries were answered automatically, because they did not concern a FP. The expert feedback was stored in the format of this clustering.

Next, the 2018 segments were predicted with the usual pipeline. We used the same segmentation score and thresholds as before.

Then, we improved these predictions using the expert feedback of the 2017 COBRAS clustering. As such the knowledge of the previous year is transferred to the current year. This result is then the improved prediction for 2018.

Note that this improved prediction is based both on the scores of the anomaly detector of 2018 and the expert knowledge on the predictions of 2017.

6. EXPERIMENTS AND RESULTS



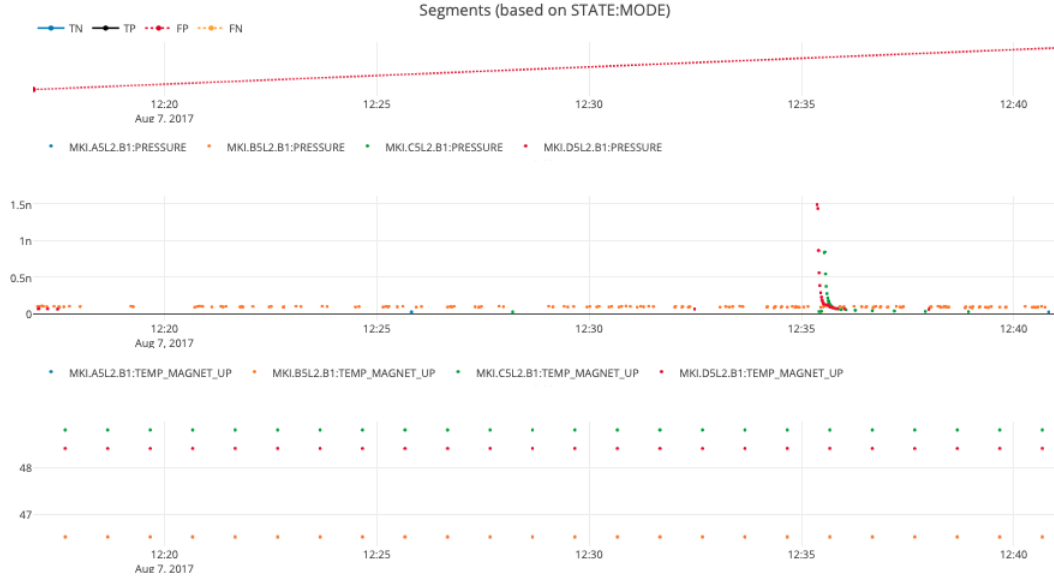
(a) Segment 103 (FP)



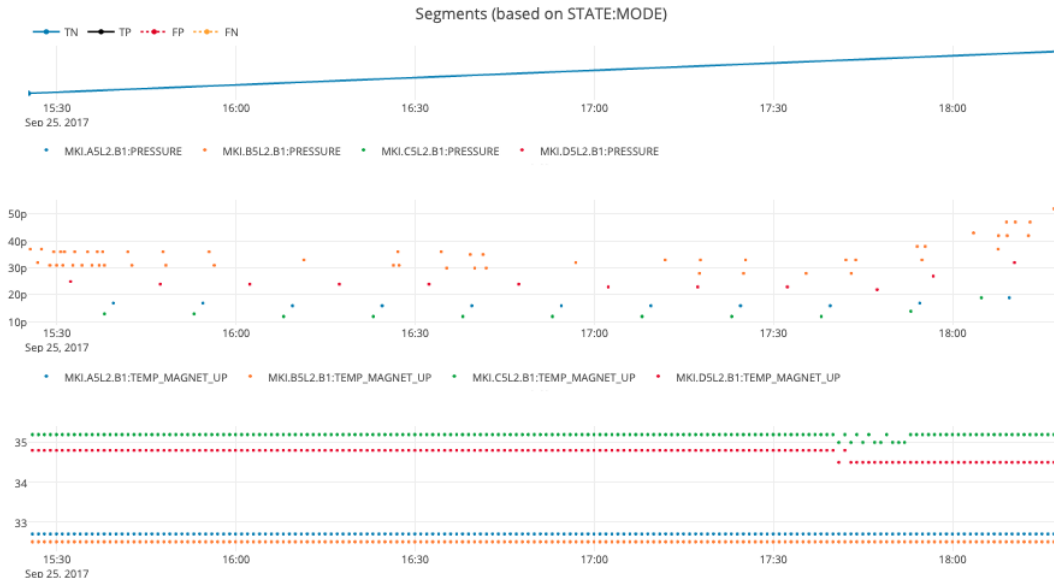
(b) Segment 1290 (TP)

Figure 6.11: Special case 1: both the FP and TP contain abnormal patterns in the pressure. The expert should answer the query with *yes*, with the aim to cluster the FP as a TP.

6.5. Incorporating expert feedback



(a) Segment 629 (FP)



(b) Segment 949 (TN)

Figure 6.12: Special case 2: the FP contains a clear spike in the pressure, whereas the TN shows normal behaviour. The expert should answer the query with *no*.

Table 6.12: Confusion matrix and metrics using the pipeline with parameters of Table 6.2, anomaly_detector=GMM, threshold=99%.

(a) 2017

	anomaly	normal
detected	5	8
undetected	3	1281

precision	recall
0.38	0.62

(b) 2018: without and with COBRAS improvement.

	anomaly	normal
detected	4	11
undetected	3	1428

	anomaly	normal
detected	6	2
undetected	1	1437

precision	recall
0.27	0.57

precision	recall
0.75	0.86

Results

GMM The confusion matrices for 2017 and 2018 (without and with improvement) are shown in Table 6.12. One can see that this method is effective. The number of FPs was reduced and two more anomalies were detected. This is the overall best result we have obtained for predicting 2018.

iForest The confusion matrices for 2017 and 2018 (without and with improvement) are shown in Table 6.13. The result is quite surprising. Although iFOREST does not succeed in making good predictions with the anomaly detector itself, in combination with COBRAS it appears to give useful predictions. Not only all FPs have disappeared, the number of FNs has decreased as well.

6.5.4 Conclusion

We have shown that by incorporating expert feedback, the number of FPs can be decreased. In one setting, there was even no single FP remaining, giving a precision of 1.00.

Furthermore, experimental results suggest that COBRAS clustering is also helpful to transform FNs into TPs. Therefore a little amount of interactive feedback by experts can positively influence the performance of the detector.

Table 6.13: Confusion matrix and metrics using the pipeline with parameters of Table 6.4, `anomaly_detector=IFOREST`, `threshold=95%`.

(a) 2017

	anomaly	normal
detected	2	63
undetected	6	1226

precision	recall
0.03	0.25

(b) 2018: without and with COBRAS improvement.

	anomaly	normal
detected	1	72
undetected	6	1367

precision	recall
0.01	0.14

	anomaly	normal
detected	3	0
undetected	4	1439

precision	recall
1.00	0.43

Chapter 7

Conclusion

The goal of this work was the development of an anomaly detection application for the injection kicker magnets of the LHC, which supports both an offline and online mode. In this last chapter we summarize our main contributions and propose future research directions.

7.1 Main contributions

In Chapter 5 an anomaly detection application has been proposed. It is structured as a pipeline which includes preprocessing, detection, postprocessing and evaluation. A final visualisation step shows the input data and resulting predictions in an interactive explorer. New state and controller variables were added as features. The data used is more recent and has a larger timespan of 3 years compared to 1.5 year before. Additionally, the feature engineering step was extended with Fourier features (frequency and intensity). Together with improved filters, these additions have led to a better model for detecting anomalies. In the postprocessing step, we proposed an exact segmentation method based on the state data. It replaces a previous method with a fixed `segmentation_distance` parameter. Our new method resulted in more meaningful predictions.

In Chapter 6 the performance of the application has been tested in several experiments. It is the first time that the application was tested extensively, with the new data.

First, a grid search has led to an optimal set of parameters per year together with good predictions. Based on these results, we proposed a set of general parameters.

Second, feature selection has led to an optimal subset of features per year for both anomaly detectors. It consists of at most 16 features for GMM and 26 features for IFOREST. Using a model trained on these subsets of features with the general parameters, the application detects the same amount of anomalies as with a model using all features with optimal parameters from a grid search. The proposed feature subsets are beneficial, because they make the training process faster and allow for more generalisation.

Third, we moved from an offline to online setting. Using the initially chosen thresholds, the results were worse, but when decreasing the threshold appropriately the model was still able to detect the same anomalies as in the offline mode (at the cost of a lower precision). However, it remains to be examined how to adjust the threshold and how to cope with the resulting FPs.

Fourth, we have proposed a method for incorporating expert feedback. In an interactive way, experts can correct predictions by clustering the predicted segments of historical data. This knowledge is then transferred to new predictions. It has led to promising results with a lower number of FPs. Surprisingly the technique also helped to increase both precision and recall of iFOREST which did not perform well without the additional expert knowledge.

Based on the results of the experiments, we have concluded that GMM outperforms iFOREST. The latter one also detects anomalies correctly but its AUC and rank metric are worse than for GMM.

7.2 Future work

Although the current application performs reasonably well, there are different open problems giving rise to new research directions.

First, the current system does not provide *interpretability*. When an anomaly is detected, it would be beneficial for CERN to indicate what exactly caused the anomaly. Exploring other anomaly detectors could be an approach to solve this problem. In addition to the Isolation Forest interpretability, the MERCS [20] model, developed by Van Wolputte et al., could be explored.

Second, the more data to be used for training and prediction, the more *computational power* will be needed. With the current amount of data it was for example unfeasible to exhaustively execute all configurations for a grid search. For future particle accelerators, the data generated during experiments will be even larger, leading to a longer execution time. However, CERN is working on a new logging system with a Spark back-end. This system allows automatic parallelisation and is developed for large-scale data operations.

Third, the interactive clustering COBRAS has shown to be useful in reducing the number of FPs. However, at this point COBRAS has only access to a limited amount of attributes. Possibly the system could work even better, by providing more data of other types. This could be tested in future work.

Finally, it is possible that the FPs are in fact no false alarms but indeed anomalies with reference to the trained model. These FPs might not have led to equipment failure but could be telltale signs of failure. The correlation of the FPs with the found anomalies should be investigated.

Appendix A

List of all data collections

Source: [8].

LHC.BCTFR.A6R4.B1:BEAM_INTENSITY	MKI.C5R8.B2:PRESSURE
LHC.BCTFR.A6R4.B2:BEAM_INTENSITY	MKI.C5R8.B2:TEMP_MAGNET_DOWN
LHC.BQM.B1:BUNCH_LENGTH_MEAN	MKI.C5R8.B2:TEMP_MAGNET_UP
LHC.BQM.B2:BUNCH_LENGTH_MEAN	MKI.C5R8.B2:TEMP_TUBE_DOWN
MKI.A5L2.B1:PRESSURE	MKI.C5R8.B2:TEMP_TUBE_UP
MKI.A5L2.B1:TEMP_MAGNET_DOWN	MKI.D5L2.B1:PRESSURE
MKI.A5L2.B1:TEMP_MAGNET_UP	MKI.D5L2.B1:TEMP_MAGNET_DOWN
MKI.A5L2.B1:TEMP_TUBE_DOWN	MKI.D5L2.B1:TEMP_MAGNET_UP
MKI.A5L2.B1:TEMP_TUBE_UP	MKI.D5L2.B1:TEMP_TUBE_DOWN
MKI.A5R8.B2:PRESSURE	MKI.D5L2.B1:TEMP_TUBE_UP
MKI.A5R8.B2:TEMP_MAGNET_DOWN	MKI.D5R8.B2:PRESSURE
MKI.A5R8.B2:TEMP_MAGNET_UP	MKI.D5R8.B2:TEMP_MAGNET_DOWN
MKI.A5R8.B2:TEMP_TUBE_DOWN	MKI.D5R8.B2:TEMP_MAGNET_UP
MKI.A5R8.B2:TEMP_TUBE_UP	MKI.D5R8.B2:TEMP_TUBE_DOWN
MKI.B5L2.B1:PRESSURE	MKI.D5R8.B2:TEMP_TUBE_UP
MKI.B5L2.B1:TEMP_MAGNET_DOWN	MKI.ELOGBOOK
MKI.B5L2.B1:TEMP_MAGNET_UP	MKI.ELOGBOOK_tagged
MKI.B5L2.B1:TEMP_TUBE_DOWN	MKI.UA23.F3.CONTROLLER:
MKI.B5L2.B1:TEMP_TUBE_UP	KICK_COUNT_TOPLAY
MKI.B5R8.B2:PRESSURE	MKI.UA23.F3.CONTROLLER:
MKI.B5R8.B2:TEMP_MAGNET_DOWN	KICK_DELAY_TOPLAY
MKI.B5R8.B2:TEMP_MAGNET_UP	MKI.UA23.F3.CONTROLLER:
MKI.B5R8.B2:TEMP_TUBE_DOWN	KICK_ENABLE_TOPLAY
MKI.B5R8.B2:TEMP_TUBE_UP	MKI.UA23.F3.CONTROLLER:
MKI.C5L2.B1:PRESSURE	KICK_LENGTH_TOPLAY
MKI.C5L2.B1:TEMP_MAGNET_DOWN	MKI.UA23.F3.CONTROLLER:
MKI.C5L2.B1:TEMP_MAGNET_UP	KICK_STRENGTH_TOPLAY
MKI.C5L2.B1:TEMP_TUBE_DOWN	MKI.UA23.F3.CONTROLLER:
MKI.C5L2.B1:TEMP_TUBE_UP	KICK_TIME_TOPLAY

A. LIST OF ALL DATA COLLECTIONS

MKI.UA23.IPOC.AB1:E_KICK	KICK_LENGTH_TOPLAY
MKI.UA23.IPOC.AB1:I_STRENGTH	MKI.UA87.F3.CONTROLLER:
MKI.UA23.IPOC.AB1:T_DELAY	KICK_STRENGTH_TOPLAY
MKI.UA23.IPOC.AB1:T_FALLTIME	MKI.UA87.F3.CONTROLLER:
MKI.UA23.IPOC.AB1:T_LENGTH	KICK_TIME_TOPLAY
MKI.UA23.IPOC.AB1:T_RISETIME	MKI.UA87.IPOC.AB2:E_KICK
MKI.UA23.IPOC.AB1:T_START_TH	MKI.UA87.IPOC.AB2:I_STRENGTH
MKI.UA23.IPOC.BB1:E_KICK	MKI.UA87.IPOC.AB2:T_DELAY
MKI.UA23.IPOC.BB1:I_STRENGTH	MKI.UA87.IPOC.AB2:T_FALLTIME
MKI.UA23.IPOC.BB1:T_DELAY	MKI.UA87.IPOC.AB2:T_LENGTH
MKI.UA23.IPOC.BB1:T_FALLTIME	MKI.UA87.IPOC.AB2:T_RISETIME
MKI.UA23.IPOC.BB1:T_LENGTH	MKI.UA87.IPOC.AB2:T_START_TH
MKI.UA23.IPOC.BB1:T_RISETIME	MKI.UA87.IPOC.BB2:E_KICK
MKI.UA23.IPOC.BB1:T_START_TH	MKI.UA87.IPOC.BB2:I_STRENGTH
MKI.UA23.IPOC.CB1:E_KICK	MKI.UA87.IPOC.BB2:T_DELAY
MKI.UA23.IPOC.CB1:I_STRENGTH	MKI.UA87.IPOC.BB2:T_FALLTIME
MKI.UA23.IPOC.CB1:T_DELAY	MKI.UA87.IPOC.BB2:T_LENGTH
MKI.UA23.IPOC.CB1:T_FALLTIME	MKI.UA87.IPOC.BB2:T_RISETIME
MKI.UA23.IPOC.CB1:T_LENGTH	MKI.UA87.IPOC.BB2:T_START_TH
MKI.UA23.IPOC.CB1:T_RISETIME	MKI.UA87.IPOC.CB2:E_KICK
MKI.UA23.IPOC.CB1:T_START_TH	MKI.UA87.IPOC.CB2:I_STRENGTH
MKI.UA23.IPOC.DB1:E_KICK	MKI.UA87.IPOC.CB2:T_DELAY
MKI.UA23.IPOC.DB1:I_STRENGTH	MKI.UA87.IPOC.CB2:T_FALLTIME
MKI.UA23.IPOC.DB1:T_DELAY	MKI.UA87.IPOC.CB2:T_LENGTH
MKI.UA23.IPOC.DB1:T_FALLTIME	MKI.UA87.IPOC.CB2:T_RISETIME
MKI.UA23.IPOC.DB1:T_LENGTH	MKI.UA87.IPOC.CB2:T_START_TH
MKI.UA23.IPOC.DB1:T_RISETIME	MKI.UA87.IPOC.DB2:E_KICK
MKI.UA23.IPOC.DB1:T_START_TH	MKI.UA87.IPOC.DB2:I_STRENGTH
MKI.UA23.STATE:CONTROL	MKI.UA87.IPOC.DB2:T_DELAY
MKI.UA23.STATE:MODE	MKI.UA87.IPOC.DB2:T_FALLTIME
MKI.UA23.STATE:SOFTSTARTSTATE	MKI.UA87.IPOC.DB2:T_LENGTH
MKI.UA23.STATE:STATUS	MKI.UA87.IPOC.DB2:T_RISETIME
MKI.UA87.F3.CONTROLLER:	MKI.UA87.IPOC.DB2:T_START_TH
KICK_COUNT_TOPLAY	MKI.UA87.STATE:CONTROL
MKI.UA87.F3.CONTROLLER:	MKI.UA87.STATE:MODE
KICK_DELAY_TOPLAY	MKI.UA87.STATE:SOFTSTARTSTATE
MKI.UA87.F3.CONTROLLER:	MKI.UA87.STATE:STATUS
KICK_ENABLE_TOPLAY	
MKI.UA87.F3.CONTROLLER:	

Appendix B

Feature selection: feature sets

B.1 GMM

Listing B.1: Optimal set of features after 20 iterations of feature selection for GMM (2016).

```
# Execution time: 3 hours, 54 minutes, 38 seconds
2 'MKI.UA23.IPOC.CB1:T_DELAY',
  'MKI.UA23.STATE:CONTROL',
4 'MKI.B5L2.B1:PRESSURE:SW_MEAN_DIFF_600_s',
  'MKI.D5L2.B1:PRESSURE:SW_MEAN_DIFF_600_s',
6 'MKI.UA23.IPOC.DB1:T_START_TH',
  'MKI.D5L2.B1:TEMP_MAGNET_UP_fft_freq_max',
8 'MKI.D5L2.B1:TEMP_MAGNET_DOWN',
  'MKI.C5L2.B1:TEMP_MAGNET_UP',
10 'MKI.UA23.IPOC.DB1:T_FALLTIME',
  'MKI.D5L2.B1:TEMP_MAGNET_DOWN_fft_int_max_all',
12 'MKI.A5L2.B1:PRESSURE_fft_int_max_all',
  'MKI.A5L2.B1:TEMP_MAGNET_UP_fft_freq_percentage_intensity5'
```

Listing B.2: Optimal set of features after 20 iterations of feature selection for GMM (2017).

```
1 # Execution time: 6 hours, 59 minutes, 39 seconds
  'MKI.UA23.IPOC.CB1:T_DELAY',
3 'MKI.UA23.STATE:SOFTSTARTSTATE',
  'MKI.UA23.STATE:CONTROL',
5 'MKI.B5L2.B1:TEMP_TUBE_UP',
  'MKI.C5L2.B1:TEMP_TUBE_DOWN_fft_freq_max',
7 'MKI.D5L2.B1:PRESSURE',
  'MKI.A5L2.B1:TEMP_MAGNET_DOWN:SW_MEAN_DIFF_600_s',
9 'MKI.A5L2.B1:TEMP_TUBE_UP:SW_MEAN_DIFF_600_s',
  'MKI.C5L2.B1:PRESSURE',
```

B. FEATURE SELECTION: FEATURE SETS

```
11 'MKI.B5L2.B1:TEMP_TUBE_UP_fft_freq_max',  
    'MKI.B5L2.B1:TEMP_MAGNET_UP_fft_freq_percentage_intensity20',  
13 'MKI.A5L2.B1:TEMP_MAGNET_UP_fft_freq_max'
```

Listing B.3: Optimal set of features after 20 iterations of feature selection for GMM (2018).

```
1 #Execution time: 4 hours, 20 minutes, 52 seconds  
  'MKI.A5L2.B1:TEMP_TUBE_DOWN_fft_freq_max',  
3  'MKI.UA23.IPOC.CB1:T_DELAY',  
  'MKI.UA23.STATE:SOFTSTARTSTATE',  
5  'MKI.UA23.F3.CONTROLLER:KICK_LENGTH_TOPLAY',  
  'MKI.UA23.STATE:CONTROL',  
7  'MKI.B5L2.B1:PRESSURE:SW_MEAN_DIFF_600_s',  
  'MKI.B5L2.B1:TEMP_TUBE_UP',  
9  'MKI.D5L2.B1:TEMP_MAGNET_UP:SW_MEAN_DIFF_600_s',  
  'MKI.A5L2.B1:TEMP_MAGNET_UP:SW_MEAN_DIFF_600_s',  
11 'MKI.A5L2.B1:PRESSURE_fft_freq_percentage_intensity5',  
  'MKI.C5L2.B1:PRESSURE_fft_int_median',  
13 'MKI.B5L2.B1:TEMP_TUBE_DOWN_fft_freq_max',  
  'MKI.B5L2.B1:PRESSURE_fft_freq_max',  
15 'MKI.C5L2.B1:TEMP_TUBE_DOWN:SW_MEAN_DIFF_600_s',  
  'MKI.B5L2.B1:TEMP_MAGNET_UP_fft_freq_percentage_intensity5',  
17 'MKI.A5L2.B1:PRESSURE:SW_MEAN_DIFF_600_s'
```

B.2 iForest

Listing B.4: Optimal set of features after 20 iterations of feature selection for iFOREST (2016).

```
1 # Execution time: 7 hours, 38 minutes, 13 seconds  
  'MKI.UA23.IPOC.CB1:T_DELAY',  
3  'MKI.UA23.STATE:CONTROL',  
  'MKI.B5L2.B1:TEMP_TUBE_UP',  
5  'MKI.B5L2.B1:TEMP_TUBE_UP:SW_MEAN_DIFF_600_s',  
  'MKI.D5L2.B1:TEMP_MAGNET_DOWN:SW_MEAN_DIFF_600_s',  
7  'MKI.A5L2.B1:TEMP_MAGNET_UP_fft_int_max_all',  
  'MKI.D5L2.B1:TEMP_TUBE_DOWN_fft_int_max_all',  
9  'MKI.C5L2.B1:TEMP_TUBE_UP_fft_freq_percentage_intensity10',  
  'MKI.D5L2.B1:PRESSURE',  
11 'MKI.D5L2.B1:PRESSURE_fft_freq_percentage_intensity5',  
  'MKI.UA23.IPOC.AB1:I_STRENGTH',  
13 'MKI.UA23.IPOC.AB1:E_KICK',  
  'MKI.B5L2.B1:PRESSURE_fft_freq_percentage_intensity10',
```

```

15 'MKI.D5L2.B1:TEMP_TUBE_DOWN:SW_SUM_600_s',
    'MKI.C5L2.B1:TEMP_MAGNET_DOWN_fft_int_median',
17 'MKI.D5L2.B1:TEMP_TUBE_UP_fft_int_median'

```

Listing B.5: Optimal set of features after 20 iterations of feature selection for iFOREST (2017).

```

1  # Execution time: 4 hours, 41 minutes, 6 seconds
    'MKI.UA23.IPOC.CB1:T_DELAY',
3  'LHC.BCTFR.A6R4.B1:BEAM_INTENSITY',
    'MKI.UA23.STATE:SOFTSTARTSTATE',
5  'MKI.UA23.F3.CONTROLLER:KICK_LENGTH_TOPLAY',
    'MKI.UA23.STATE:CONTROL',
7  'MKI.B5L2.B1:PRESSURE:SW_MEAN_DIFF_600_s',
    'MKI.B5L2.B1:TEMP_TUBE_UP',
9  'MKI.B5L2.B1:TEMP_TUBE_UP:SW_MEAN_DIFF_600_s',
    'MKI.UA23.IPOC.DB1:I_STRENGTH',
11 'MKI.C5L2.B1:PRESSURE',
    'MKI.UA23.IPOC.BB1:T_START_TH',
13 'MKI.D5L2.B1:TEMP_TUBE_DOWN_fft_freq_percentage_intensity10',
    'MKI.UA23.F3.CONTROLLER:KICK_TIME_TOPLAY',
15 'MKI.UA23.IPOC.BB1:I_STRENGTH',
    'MKI.C5L2.B1:TEMP_TUBE_UP_fft_int_max_all',
17 'MKI.A5L2.B1:TEMP_MAGNET_DOWN:SW_MEAN_DIFF_600_s',
    'MKI.UA23.IPOC.AB1:I_STRENGTH',
19 'MKI.A5L2.B1:PRESSURE_fft_int_median',
    'MKI.UA23.IPOC.DB1:T_LENGTH',
21 'MKI.UA23.IPOC.DB1:T_DELAY',
    'MKI.C5L2.B1:TEMP_TUBE_UP_fft_int_median',
23 'MKI.D5L2.B1:TEMP_TUBE_DOWN_fft_freq_max',
    'MKI.UA23.IPOC.AB1:T_LENGTH',
25 'MKI.D5L2.B1:TEMP_TUBE_DOWN_fft_freq_percentage_intensity20',
    'MKI.C5L2.B1:TEMP_TUBE_UP',
27 'LHC.BQM.B1:BUNCH_LENGTH_MEAN:SW_MEAN_DIFF_600_s'

```

Listing B.6: Optimal set of features after 20 iterations of feature selection for iFOREST (2018).

```

1  # Execution time: 11 hours, 14 minutes, 36 seconds (4-core)
    'MKI.C5L2.B1:TEMP_MAGNET_UP_fft_freq_max',
3  'MKI.A5L2.B1:TEMP_TUBE_DOWN_fft_freq_max',
    'MKI.UA23.IPOC.CB1:T_DELAY',
5  'MKI.UA23.STATE:SOFTSTARTSTATE',
    'MKI.UA23.STATE:CONTROL',

```

B. FEATURE SELECTION: FEATURE SETS

```
7 'MKI.B5L2.B1:PRESSURE:SW_MEAN_DIFF_600_s',
  'MKI.B5L2.B1:TEMP_TUBE_UP:SW_MEAN_DIFF_600_s',
9 'MKI.D5L2.B1:TEMP_TUBE_DOWN_fft_freq_percentage_intensity5',
  'MKI.B5L2.B1:TEMP_TUBE_DOWN_fft_freq_percentage_intensity10',
11 'MKI.C5L2.B1:TEMP_MAGNET_UP_fft_int_median',
  'MKI.A5L2.B1:TEMP_MAGNET_DOWN_fft_int_max_all',
13 'MKI.B5L2.B1:TEMP_TUBE_DOWN_fft_freq_percentage_intensity20',
  'MKI.UA23.IPOC.CB1:T_FALLTIME',
15 'MKI.C5L2.B1:PRESSURE:SW_MEAN_DIFF_600_s',
  'MKI.A5L2.B1:TEMP_TUBE_DOWN_fft_freq_percentage_intensity10',
17 'MKI.A5L2.B1:TEMP_TUBE_DOWN_fft_int_median',
  'MKI.D5L2.B1:TEMP_TUBE_DOWN_fft_freq_percentage_intensity10',
19 'MKI.C5L2.B1:TEMP_TUBE_UP_fft_freq_percentage_intensity10',
  'MKI.B5L2.B1:TEMP_TUBE_DOWN_fft_freq_percentage_intensity5',
21 'MKI.A5L2.B1:TEMP_MAGNET_DOWN:SW_SUM_600_s',
  'MKI.A5L2.B1:TEMP_MAGNET_DOWN_fft_int_max_var',
23 'MKI.A5L2.B1:TEMP_MAGNET_DOWN_fft_freq_max',
  'MKI.UA23.IPOC.BB1:I_STRENGTH',
25 'MKI.A5L2.B1:TEMP_MAGNET_DOWN'
```

Appendix C

Poster

Anomaly detection for injection magnets at CERN

Thiebout Dewitte¹, Elia Van Wolputte², Pieter Van Trappen³, Wannes Meert², Hendrik Blockeel²

¹thiebout.dewitte@student.kuleuven.be, ²{firstname}.{lastname}@kuleuven.be, ³pieter.van.trappen@cern.ch

INTRODUCTION

Context and problem

Injection Kicker Magnets (MKI) inject protons into the Large Hadron Collider at CERN.

Manually detecting and analyzing anomalies in the MKI data is a tedious process.



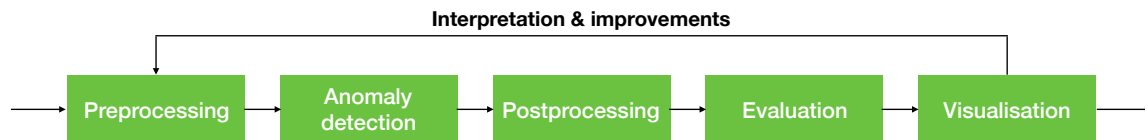
Goal

Develop an anomaly detection application which automatically detects anomalies, based on historical data.

Main challenges

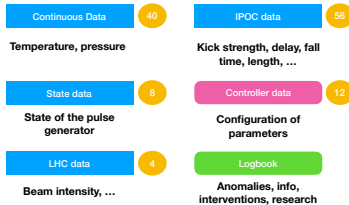
- How to merge data from different **asynchronous sources**?
- How to incorporate **noisy labels** from (few) anomalies?
- How to **evaluate** the output of the application?
- How to move from offline analysis on historical data to **live simulation**?

PIPELINE



1. Preprocessing

- Design **filters** to remove wrong measurements
- Extract **temporal features** using sliding windows and Fourier transform
- Obtain a **feature vector** by merging data of asynchronous sources



2. Anomaly detection

- Easily plug-in any detector which generates **anomaly scores** between 0 and 1
- Currently focussed on Gaussian Mixture Model and Isolation Forest

Gaussian Mixture Model

Fit the data to a fixed number of Gaussian distributions with unknown parameters.

An anomaly has a low chance to belong to any of the components.

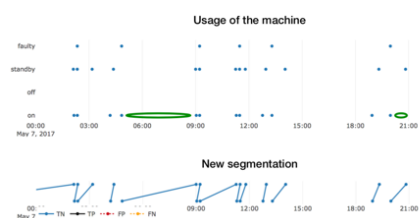
Isolation Forest

Learn an ensemble of isolation trees. This is a tree structure which tries to isolate every individual point.

An anomaly is easy to isolate from the other data points.

3. Postprocessing

- The detector returns an anomaly score for each feature vector
- However, we want to know whether an anomaly occurred during the **usage** of the machine, so we combine vectors in a **segment** based on the machine usage



4. Evaluation

- The anomaly score of a segment, is the average of the **top k** scores of its feature vectors
- Use a **threshold** to transform anomaly scores into predictions
- Use the **anomaly labels** from the **logbook** as a ground truth with a 12-hours range
- Metric: **precision** and **recall**
- For hyper-optimisation: area under precision recall curve (**AUC**)

5. Visualisation

- Live validation metrics** (threshold, segment score method, ...)
- Which anomalies did the application **miss**?
- Why** was a segment predicted as anomalous?



Results for 3 months (GMM, 98.5% threshold)

Historical data

	Anomaly	Normal
Detected	5	4
Undetected	3	567

Precision: 0.56
Recall: 0.62

Live simulation per week

	Anomaly	Normal
Detected	4	9
Undetected	4	551

Precision: 0.23
Recall: 0.38

Bibliography

- [1] R. Barlow, M. Laffin, M. Barnes, and E. Carlier. KISS conditioning of the LHC injection kickers. Technical report, 2010.
- [2] M. J. Barnes, L. Ducimetiere, T. Fowler, V. Senaj, and L. Sermeus. Injection and extraction magnets: kicker magnets. 2011.
- [3] R. Billen and C. Roderick. The LHC Logging Service: Capturing, storing and using time-series data for the world’s largest scientific instrument. Technical Report AB-Note-2006-046. CERN-AB-Note-2006-046, CERN, Nov 2006.
- [4] CERN. The Large Hadron Collider. <https://home.cern/science/accelerators/large-hadron-collider/>, 2018. [Online; accessed 21-December-2018].
- [5] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3), 2009.
- [6] V. E. Fortov, B. Goddard, D. H. H. Hoffmann, V. Kain, I. V. Lomonosov, A. R. Piriz, R. Schmidt, A. Shutov, N. A. Tahir, and M. Temporal. Interaction of the CERN Large Hadron Collider (LHC) Beam with Solid Metallic Targets. (LHC-Project-Report-748. CERN-LHC-Project-Report-748):4 p, 2004. revised version submitted on 2004-09-23 10:53:07.
- [7] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.
- [8] A. Halilovic. Anomaly Detection for the CERN Large Hadron Collider injection magnets, 2018.
- [9] P. T. Inc. Collaborative data science, 2015.
- [10] W. J. Jurasz. NXCALS - big data logging system - deployment and monitoring. 4th Developers@CERN Forum. Oct 2017.
- [11] F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE, 2008.
- [12] H. Liu and H. Motoda. *Feature extraction, construction and selection: A data mining perspective*, volume 453. Springer Science & Business Media, 1998.

- [13] N. Magnin, E. Carlier, B. Goddard, V. Mertens, and J. A. Uythoven. Internal Post Operation Check System for Kicker Magnet Current Waveforms Surveillance. Mar 2014.
- [14] E. Mobs. The CERN accelerator complex - August 2018. Aug 2018. General Photo.
- [15] A. Pantelia. General Activities at the tunnel during LS1. Jan 2014. General Photo.
- [16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [17] G. Schwarz et al. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978.
- [18] T. van Craenendonck, S. Dumancic, E. V. Wolputte, and H. Blockeel. COBRAS: Fast, Iterative, Active Clustering with Pairwise Constraints. *CoRR*, abs/1803.11060, 2018.
- [19] T. Van Craenendonck, W. Meert, S. Dumančić, and H. Blockeel. COBRAS TS: A New Approach to Semi-supervised Clustering of Time Series. In *International Conference on Discovery Science*, pages 179–193. Springer, 2018.
- [20] E. Van Wolputte, E. Korneva, and H. Blockeel. MERCS: multi-directional ensembles of regression and classification trees. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [21] N. Wéry. Anomalie detectie voor de Large Hadron Collider, 2017.

Fiche masterproef

Student: Thiebout Dewitte

Titel: Anomaly detection for the injection kicker magnets of the CERN Large Hadron Collider

Nederlandse titel: Anomalie detectie voor de injectie kicker magneten van de CERN Large Hadron Collider

UDC: 681.3

Korte inhoud:

Reliability, availability and maintainability are parameters that determine if a large-scale accelerator system can be operated in a sustainable and cost effective manner. Beam transfer equipment such as kicker systems are critical components with potential significant impact on the global performance of the entire machine complex. Identifying root causes of malfunctions is currently tedious, and will become infeasible in future systems due to increasing complexity. Machine learning can help to automate this process. For this purpose a collaboration between CERN and KU Leuven was founded. Several iterations of the study have yielded an *anomaly detection pipeline* which includes preprocessing, detection, postprocessing and evaluation. Merging data of different, asynchronous sources is one of the main challenges. Currently, Gaussian Mixture Models and Isolation Forests are used as unsupervised detectors, but any detector can easily be plugged into the system. During evaluation, we compare the detector predictions to manual e-logbook entries, which constitute a noisy ground truth. Lastly, we try to incorporate expert knowledge by means of semi-supervised clustering with COBRAS. A grid search allows for hyper-parameter optimization across the entire pipeline, yielding promising results.

Thesis voorgedragen tot het behalen van de graad van Master of Science in de ingenieurswetenschappen: computerwetenschappen, hoofdoptie Computacionele informatica

Promotoren: Prof. dr. ir. H. Blockeel
Dr. ir. W. Meert

Assessoren: E. Van Wolputte
Prof. dr. D. Hughes

Begeleiders: E. Van Wolputte
Ing. P. Van Trappen