KU LEUVEN

FACULTEIT INGENIEURSWETENSCHAPPEN

Anomaly Detection for the CERN Large Hadron Collider injection magnets

Armin Halilovic

CERN-THESIS-2018-372
28/06/2018

Thesis submitted for the degree of Master of Science in Engineering: Computer Science, option Artificial Intelligence

Thesis supervisors:
Prof. Dr. ir. Hendrik Blockeel
Dr. ir. Wannes Meert

Assessors:
Prof. Dr. Bruno Crispo
Dr. Stefano Teso

Mentors:
Elia Van Wolputte
Ir. Pieter Van Trappen

Academic year 2017 – 2018

Preface

First of all, I would like to thank my mentor, Elia van Wolputte, for his much needed guidance throughout this thesis. Many thanks to the contact person at CERN, Pieter van Trappen, as well, who was always ready to answer my questions and provide me with tons of data and information on the workings of the Large Hadron Collider.

I would also like to express my gratitude towards the promotors of this thesis, Hendrik Blockeel and Wannes Meert, for their very useful advice on the contacts moments we had.

Lastly, I extend my thanks to CERN, for providing the unique opportunity to work with LHC machine data. It was an extremely interesting experience to work on a project of this scale.

Armin Halilovic

Contents

Abstract

The Large Hadron Collider is the world's largest single machine and the most powerful particle accelerator ever built. Inside it, high-energy particle beams are made to collide at speeds near the speed of light. The results of the collisions are then analysed in the hope that some of the fundamental open questions in physics can be answered.

Particle beams are injected into the LHC and guided throughout it by thousands of electromagnets. New state of the art equipment for the LHC was developed by CERN in collaboration with thousands of scientists and engineers all over the world. The result of this is that, due to yet undiscovered reasons, equipment occasionally behaves in unexpected ways. This can cause experiments to fail, because of which the particle beams must then be ejected from the LHC so that the whole process of creating high-energy particle beams can be restarted. The behaviour that causes this is called anomalous behaviour. The goal of this thesis is to develop a method that can detect anomalies in the behaviour of injection kicker magnets, the magnets that inject particles into the LHC.

A system that detects anomalies cannot be built manually, as there are over a hundred data signals related to the magnets and many complex relationships exist between the signals. Currently, anomalies are only detected at CERN after they have caused experiments to fail. A machine learning based application that can learn the complex relationships between signals and that can then detect anomalies automatically would thus greatly benefit CERN. Reaction times to anomalies could be improved and less time would need to be spent meticulously analysing signal data to find the causes of anomalies. Furthermore, it is possible for machine learning models to list the factors that contribute most to anomalous behaviour. Such information could be used by CERN to improve equipment.

This thesis proposes an extensible anomaly detection application which supports the use of many different machine learning models. There is a limited amount of unspecific knowledge about when and where anomalies occurred in injection kicker magnet behaviour. A novel evaluation procedure is introduced that can use this unspecific knowledge to measure and compare anomaly detection performance fairly.

List of Figures and Tables

List of Figures

List of Tables

List of Abbreviations and Symbols

Abbreviations

BIS	Beam Interlock System
CALS	CERN Accelerator Logging Service
CERN	European Organization for Nuclear Research
CCC	CERN Control Centre
GMM	Gaussian Mixture Model
IPOC	Internal Post Operational Check
KISS	Kicker Injection Soft Start
LHC	Large Hadron Collider
MKI	Kicker injection magnets
MD	Machine Development
NaN	Not a number
PR	Precision-Recall
ROC	Receiver Operating Characteristic
SPS	Super Proton Synchrotron
SVM	Support Vector Machine

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 $1.6 * 10^{-19}$ joules
µs	microsecond
ns	nanosecond
V	volt, unit of electric potential

Chapter

1 Introduction

This thesis introduces an approach to the automated detection of anomalies in the behaviour of certain components of the Large Hadron Collider (LHC). The LHC is the world's largest and most powerful particle accelerator. Inside of it, high energy particle beams are being collided so that the results of the collisions can be analysed in state of the art particle physics research. The LHC is an enormous structure that consists of thousands of magnets that guide particles through their correct paths. This thesis focuses on a handful of these magnets, the injection kicker magnets. The injection kickers are responsible for the injection of particles into the LHC.

Anomalous behaviour in the injection kicker magnets can cause experiments to fail. When this happens, experiments need to restart, which is a costly procedure, and a lot a data needs to be examined by people in order to figure out what caused the experiment to fail. This wastes a lot of time that could have been used for new physics experiments. Particle beams have to be extracted from the LHC and magnets need to cool down before experiments can be restarted.

Anomaly detection can provide a solution to these kinds of problems. An application that can automatically detect anomalous behaviour when it occurs will improve CERN's reaction time to anomalies, which could eventually lead to the prevention of experiments failing. If such an application can also point out why certain behaviour was believed to be anomalous, it could be used to improve the designs of the new state of the art components that were developed specifically for the LHC. This thesis proposes an extensible anomaly detection application which supports the use of many different machine learning models.

This thesis builds further on the work done by N. Wéry during academic year 2016–2017 [24]. In his work, Wéry used a Gaussian Mixture Model to do anomaly detection. The results ended up being unsatisfactory; the majority of known anomalies were not detected. A few of Wéry's ideas have been reused for this thesis and comparisons between the methods that were developed are made throughout this text.

The main difference between the two approaches is that Wéry's application has been transformed into a full-fledged machine learning pipeline with four distinct steps: preprocessing, anomaly detection, postprocessing, and evaluation. The pipeline is discussed in great detail in chapter 5. Each of the pipeline components has been designed to be configurable through various parameters and to be extensible by new

implementation of methods. For example, the anomaly detection step can be handled by any machine learning model that implements a certain given interface.

In order to be able to evaluate and correctly compare the performance achieved by different anomaly detectors and parameter settings, a novel evaluation strategy had to be developed. It is described in section 5.3. This was necessary because of the uncertainty of labeled data (subsection 4.1.6).

Finally, experiments were executed with two such models, Gaussian Mixture Model and Isolation Forest, and a wide range of parameter values. The results are discussed in chapter 6.

Chapter

2 Background

2.1 Large Hadron Collider

The Large Hadron Collider is the world's largest and most powerful particle accelerator and is being used for new particle physics research. Inside the accelerator, high-energy particles are sped up to near the speed of light, after which they are made to collide. Protons are brought into collisions at an energy of 14 TeV, heavy ions at 1148 TeV [11]. These beams are collided so that their products can be analysed by large systems of particle detectors.

The LHC is a huge machine; it is built in a circular shape out of 27km long tunnel of unique components that were developed specifically for the LHC [5]. Thousands of magnets of varying types and sizes are used to lead particle beams through correct paths in the LHC [7]. Two counter-rotating beams are injected into the LHC by a series of injector machines. The high-energy particle beams will then travel through two separate tubes, after which they can be made to collide in up to 4 interaction points.

Enormous amounts of data are generated by the LHC. Not only is data collected on physics experiments, but also on the accelerator equipment (such as the LHC) that is constantly being monitored. The monitoring of LHC components is crucial in confirming that everything is working in a safe and reliable way. For instance, CERN has implemented a system called the Beam Interlock System (BIS) [21] that relies on the monitoring system to prevent damage in the LHC by safely stopping beam operation when something goes wrong.

2.1.1 CERN's Accelerator Complex

The LHC is not the only particle accelerator at CERN [6]. Since every accelerator has a limited dynamic range, a chain of accelerators is created in order to make particle beams reach high energies. Each accelerator increases the energy of particles until they reach the maximum energy achievable by the accelerator. Once the particles reach this maximum energy, they are injected into the next accelerator, all the way up into the LHC. When particles enter the LHC, they attain speed near the speed of

light. Figure 2.1 shows the different particle accelerators and how they are connected to each other.

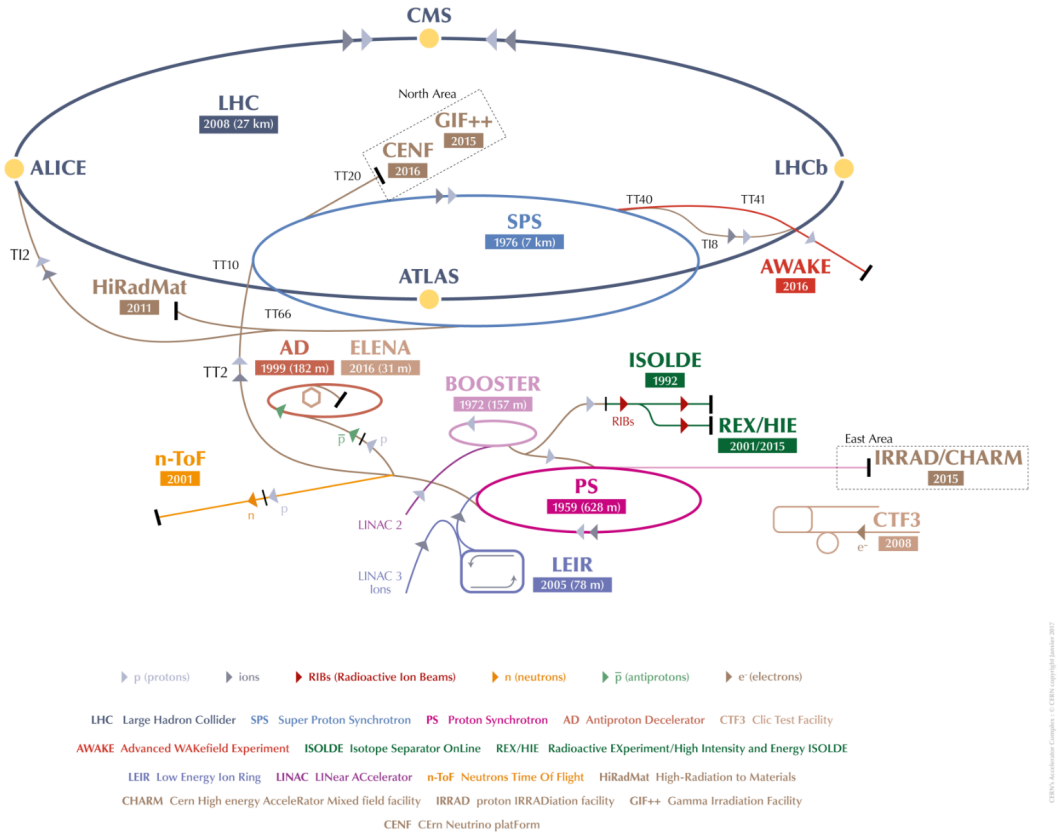


Figure 2.1: The CERN accelerator complex. Source: [17]

2.1.2 Beams

Important to note is that there are two beams in which particles travel. It is necessary to have two beams, as particles need to be accelerated in opposing directions first, before they are made to collide. The two beams are accelerated in two separate beam pipes throughout the LHC. The two beams will be referred to as beam 1 (B1) and beam 2 (B2) in the rest of this text.

During normal "LHC runs", where particle beams are injected, accelerated, and made to collide so that the results can be analysed, there will always be two beams in the LHC. It is also possible for only one beam to be present in the LHC for testing purposes. For example, during some machine development (MD) studies, only one beam is injected into the LHC so that the performance of equipment can be studied [12]. During machine development, it is also possible for two beams with different intensities to be injected. For LHC runs, however, the beams will have similar intensities in order to maximize the luminosity [13].

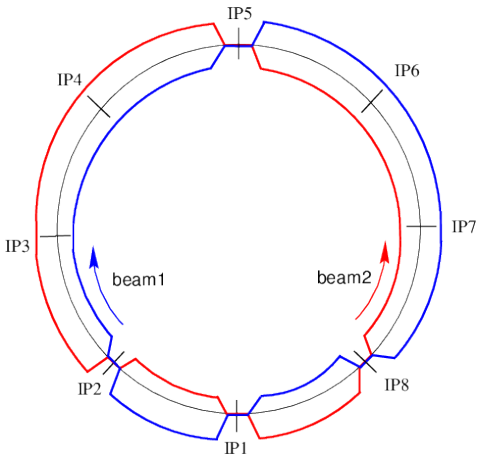


FIGURE 2.2: Schematic layout of the LHC interaction points and beams. Source: [14]

2.2 Injection Kicker Magnets

The CERN accelerator complex uses thousands of magnets for all kinds of different purposes. This study focuses on only a handful of these: the injection kicker magnets (MKI). These magnets have been designed and are maintained by the Accelerator Beam Transfer Group (ABT). They are strong electromagnets which are powered by brief pulses of electric current, which makes them produce strong pulses of magnetic fields that have a flat top duration of up to 6.6 µs [11]. Their purpose is to inject particle beams from the SPS (see figure 2.1) into the LHC. When the beams have been injected into the LHC successfully, experiments can start when collisions occur. After the injection, the MKI magnets will not be used for the rest of the LHC run. It is normal for experiments to last for several hours.

In order to inject two particle beams in opposing directions, two installations of injection kicker magnet systems have been created to handle each beam separately. Each MKI installation consists of four magnets of 2.7 m in length, named A, B, C, and D. The two systems, MKI 2 and MKI 8, are named after the interaction points they lie near (IP2 and IP8). A schematic view of the LHC interaction points is given in figure 2.2.

The fact that there are two MKI installations is important to keep in mind, as it will reveal itself in the data; there are many similar data collections where the only difference is the name of a magnet or an MKI installation. Section 4.1 provides a detailed overview of the data collections. Table 2.1 shows more information about the MKI installations. The data collections described in section 4.1 are named after a combination of the data shown in table 2.1 and the type of the measurements the collection holds. The service space in the table refers to the service tunnel next to ring that holds the pulse generator and electronics. It is separated from the main tunnel so that interference from the beams is avoided. The number 2 in "MKI 2", "5L2", "UA23" refers to interaction point 2. An analogous statement holds for interaction point 8.

Installation name	LHC beam	Ring Location	Service Space
MKI 2	Beam 1 (B1)	5L2	UA23
MKI 8	Beam 2 (B2)	5R8	UA87

Table 2.1: More information about MKI installations.

2.2.1 MKI pulse generators

Each MKI installation contains two pulse generators that power two magnets each. One generator consists of one Resonant Charging Power Supply (RCPS) and two pulse forming networks (PFNs). There is thus one PFN per magnet. PFNs can be charged up to a voltage of 54 kV, after which they can be discharged into the magnets in about 600 µs [2]. Each installation also contains a state-control and surveillance system that controls each part of the two generators of the installation.

2.2.2 KISS Conditioning

While experiments are going on in the LHC, the magnets cannot be pulsed. They would deflect the colliding beams if they did. LHC cycles last up to several hours and when the magnets cannot be used for long periods, high voltage de-conditioning[1] will occur in the magnets [1]. That affects the high voltage performance of the magnets at the next injection. Making the magnets pulse immediately at nominal voltage increases the risk of high voltage flashovers and thus deterioration of the magnets, which would cause downtime for the LHC.

To combat this risk, a process called "KISS" was developed at CERN which is executed before each injection run. KISS stands for Kicker Injection Soft Start. It reduces the risk of sparking by pre-conditioning the MKI magnets so that they can be used safely for injections. It does this by pulsing the magnets with an increasing voltage up to a voltage slightly beyond their nominal high voltage operating point. The magnets receive incrementing pulses of up to 20 000 A. The process takes about 20 minutes if no errors occur.

2.2.3 CERN Accelerator Logging Service

Each MKI installation is equipped with a bunch of sensors so that different aspects of the magnets and PFNs can be measured and stored. While these sensors can make measurements at high sampling rates, it is not feasible to store all of the measurements. There are simply so many measurements being made at CERN that storing all of them would be financially infeasible. Instead, measurements are handled in a smarter way by CALS, the CERN Accelerator Logging Service. CALS is a CERN-wide service that spans the complete accelerator complex and stores more than 100 TB/year for more than a million machine signals [3]. Instead of simply storing

[1] This de-conditioning occurs when high voltage is removed. It limits the high voltage insulation between two conductors in vacuum due to a change in the adsorbed gas layer. The de-conditioning is a function of the gas species and its pressure.

all measurements, CALS allows for only the most meaningful measurements to be stored, and different storage strategies are used different types of MKI measurements. They are explained in more depth in section 4.1. Note that CALS only handles machine data, not data generated by experiments.

2.3 Anomaly Detection

2.3.1 Classification

Anomaly detection is a classification method. The goal is to learn to distinguish abnormal from normal data. Note that abnormal data here does not necessarily mean bad or unwanted data, but also all unexpected data. The most common methods are one-class classification methods. One-class classification tries to distinguish only one type of data from all other data. On the other hand, multiclass classification separates data into multiple different classes. Multiclass anomaly detection methods exist as well, but are not explored in this thesis.

Anomaly detection methods are useful when there is not enough data to construct explicit models for all different possible behaviours. The high amount of normal data will be used by the methods to learn what normal behaviour is.

Anomaly detection can work in many different domains: Credit card transactions can be analysed for credit card fraud, MRI scans can be checked for malignant tumors, intrusion can be detected in computer network patterns, etc. In this thesis, anomaly detection will be applied on machine sensor readings to detect unexpected behaviour of machines.

2.3.2 Supervised and Unsupervised Learning

Machine learning models can be trained in different ways, two of which are supervised and unsupervised. In supervised learning, an anomaly detector would be trained to learn what constitutes anomalous behaviour by being given many examples of anomalies. In unsupervised learning, anomalous behaviour must be learned without any examples [4].

Since in the context of this thesis, there are only a small number of labels (examples) that denote when anomalies happened, the anomaly detection method developed in this thesis will rely on unsupervised learning algorithms. The fact that there are few labels is to be expected, as anomalies are few and far between.

2.3.3 Ensemble Learning

Ensemble learning is an approach in which multiple learning algorithms are trained for a task. This is done with the goal of obtaining a better combined detection performance of the ensemble than could be achieved by any of the learning algorithms themselves.

2.3.4 Anomaly Scores

In the anomaly detection application that is proposed in this thesis, machine learning models are used to assign scores to data tuples. These scores, called anomaly scores [8], represent how abnormal (how anomalous) a data tuple is. A higher anomaly score means that the data tuple is more anomalous.

2.3.5 Isolation Forest

Liu, Ting, and Zhou first introduced the Isolation Forest algorithm in 2008 [16]. It is an unsupervised tree ensemble method. It works by explicitly isolating anomalies instead of profiling normal points. The algorithm isolates data points by building simple decision trees that select features randomly and then select random split values in the ranges of the selected features. The number of splits in each tree required to isolate a sample then equals the path length from the root node to the leaf node.

Since anomalies are by definition significantly different from normal points, Isolation Forest relies on the fact that that the path length to isolate anomalous points is shorter than the path length to isolate normal points. An example of this can be seen in figure 2.3, where x_i is a normal point, and x_0 is an anomalous point. The lines in the figure display the random splits that were made before the point was isolated. The normal point requires more random splits to be isolated than the anomalous points, and will thus have a higher path length.

The path length, averaged over all such random trees in the isolation forest, is used in the decision function that generates anomaly scores for data.

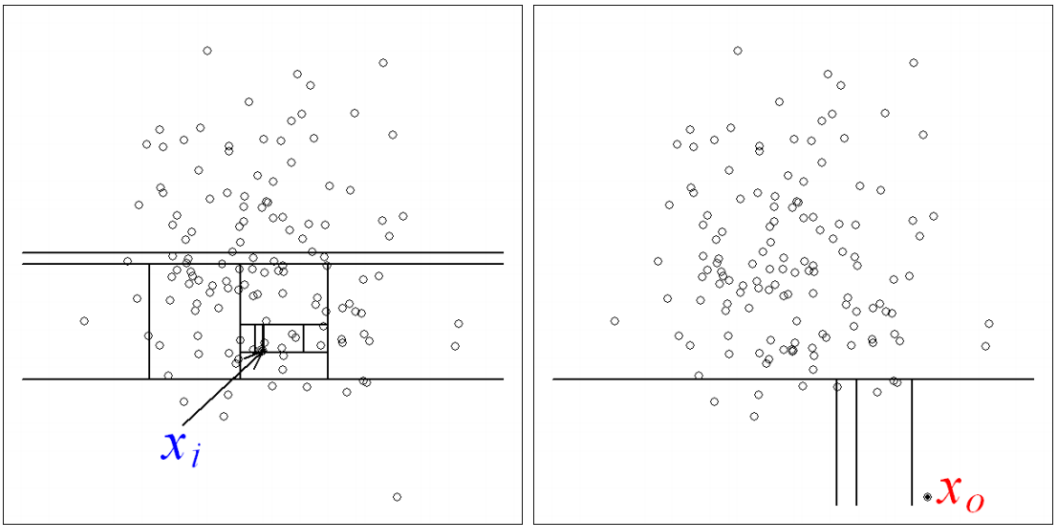


Figure 2.3: Isolation of points x_i and x_o. Source: [16]

2.4 Evaluation

In order to be able to develop a well performing anomaly detection application, a formal evaluation procedure must be set up so that the performance of different approaches can be calculated and compared correctly. A few terms and widely used classification performance metrics are introduced in this section. The best metric to use in the context of this thesis is discussed in section 5.4.2.

The metrics introduced in this section rely on the following terms: True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN). Their meanings are defined in table 2.2. The terms Positive and Negative in this context refer to the two types of behaviour that are being distinguished from each other, Positive referring to anomalous, and Negative referring to normal behaviour.

		Ground Truth	
		Positive	Negative
Prediction	Positive	TP	FP
	Negative	FN	TN

TABLE 2.2: A confusion matrix that visually defines TP, FP, FN, and TN.

Anomaly scores will be represented by numbers between 0 and 1. For these scores, a threshold must be chosen so that it can be decided which data tuples will be predicted to be anomalous (Positive). For different thresholds, there will be different numbers of TPs, FPs, FNs, and TNs, and thus also different values for performance metrics.

2.4.1 Receiver Operating Characteristics Diagrams

ROC diagrams represent a classifier's performance by creating a plot of the False Positive Rates (FPR) versus the True Positive Rates (TPR) produced by the classifier for different thresholds. The definitions of TPR and FPR are given below. The goal when using ROC diagrams as an evaluation metric is to maximise the area under the ROC curve.

$$TPR = \frac{TP}{TP + FN} \tag{2.1}$$

$$FPR = \frac{FP}{FP + TN} \tag{2.2}$$

2.4.2 Precision and Recall

Precision and Recall are defined as follows. Note that recall is the same as TPR.

$$Precision = \frac{TP}{TP + FP} \tag{2.3}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.4}$$

A more intuitive explanation of Precision and Recall is given by "scikit-learn" [18] as follows: *"Intuitively, precision is the ability of the classifier not to label as positive a sample that is negative, and recall is the ability of the classifier to find all the positive samples."*

Similarly to how TPRs and FPRs are plotted in ROC diagrams, diagrams of Precision and Recall values can be made. These are called PR curves. They are drawn by calculating Precision and Recall for a set of thresholds and creating a graph of the (precision, recall) tuples. The goal is then to maximize the area under this curve as well.

2.4.3 F Score

The F score is a measure that combines Precision and Recall into one metric. It does this by taking the harmonic average of the two.

$$F = 2 \times \frac{precision \times recall}{precision + recall} \tag{2.5}$$

Chapter

3 Problem Statement and Related Work

3.1 Motivation

The development of a successful anomaly detection application would be very useful to CERN. Anomalous behaviour in the magnets can cause physics experiments to fail, which results in a lot of lost time. Particle beams must be ejected from the LHC, magnets need to cool down before they can be used to reinject the next particle beam, etc.

Currently, when undesirable behaviour like this occurs, employees at CERN have to manually examine all magnet data to determine what exactly went wrong. This is a very time consuming and tedious process, as there are over a hundred datasets to go through. With anomaly detection, this kind of unwanted behaviour could be detected automatically. It is also possible for an anomaly detection application to even find new anomalous behaviour in past data that was not detected by CERN.

Another aspect that can be improved by anomaly detection is the reliability of the the LHC's equipment. This depends on the ability of the application to provide a reasoning about why it detects anomalous behaviour, which is called interpretability. The application could then simply point to a specific subset of data as a suspect for the anomalous behaviour. This information could then be used by CERN to improve the designs of LHC components. Isolation Forests, for instance, are interpretable, since the isolation trees could be examined to figure out which factors play the biggest roles in anomalies. GMMs, on the other hand, cannot provide such information.

3.2 Problem Statement

The LHC is a colossal machine that consists of thousands of electromagnets and generates tremendous amounts of data. It is important to note that this thesis focuses on a small subset of that data. All the data that is generated during physics experiments that serves for new research is irrelevant here. The data that this thesis focuses on is the data from the LHC injection kicker magnets. This thesis attempts to automate the detection of anomalous behaviour of the magnets. The formal problem statement of this study goes as follows:

> The goal is to develop an anomaly detection application that can detect anomalies in the behaviour of the injection kicker magnets of the Large Hadron Collider.

It is also important to note that these magnets play a relatively short role in the execution of the physics experiments in the LHC. It is normal for experiments to last for up to twelve hours. The kicker injection magnets only serve to set up these experiments by injecting particle beams into the LHC. If everything proceeds successfully, the injection phase takes about fifteen to thirty minutes. This has the result that a lot of data will be left out of consideration, because all the data generated when the magnets were not in use is not descriptive of their behaviour.

3.3 Related work

Anomaly Detection

Many different approaches to anomaly detection have been explored. Two survey papers provide a broad overview of different classes of anomaly detection methods.

First, in 2009 [8], Chandola, Banerjee, and Kumar, have laid out a structured and comprehensive overview of the research done on anomaly detection. They have grouped many methods into different categories based on the underlying approach adopted by each method. Then, for each category, they have identified advantages and disadvantages of techniques belonging to each category.

Later, in 2014 [20], Pimentel, D. Clifton, L. Clifton, and Tarassenko provided an updated investigation of anomaly detection which was based more on a signal processing point of view.

The categories of anomaly detection techniques presented by Pimentel et al. are:

- **Probabilistic:** These methods often estimate the density of "normal" behaviour. Low density areas are then assumed to contain little normal behaviour. Data tuples that fall in these are are consequently assigned high anomaly scores. Examples of probabilistic techniques are Isolation Forest and Gaussian Mixture Model, both of which are used in this thesis.
- **Distance-based:** Distance-based techniques work similarly to clustering and nearest-neighbor approaches. Normal behaviour is expected to be tightly clustered, anomalous behavior is not. Therefore, data tuples that lie farther away from their neighbors than most data tuples are more likely to be assigned higher anomaly scores. Examples are k-nearest neighbour and Local Outlier Factor.
- **Reconstruction-based:** Reconstruction-based methods rely on regression models. Such models are used to predict future data tuples. The error between the prediction and the actual observation is then used to calculate the anomaly score. Neural Networks can be used in this style of approach.
- **Domain-based:** These methods define domains that contain normal behaviour by defining a boundaries around normal behaviour based on the structure of the training dataset. For example, one-class Support Vector Machines can be

trained to determine the location of the boundaries around normal behavior using only normal data. Anomalous data is not included in the training dataset. New data can then be scored using the learned boundaries.

- **Information Theoretic:** This approach uses information-theoretic measures to calculate information content in data. It relies on the assumption that anomalous behaviour will cause the information content to change significantly. Examples are entropy and Kolmogorov complexity.

The data to be analysed in this thesis makes many techniques invalid options. To briefly introduce the data: it contains a mix of continuous and categorical measurement collections. In addition, some of the measurements are sampled at certain triggers, others use a mix of sampling rates. Also, the labeled data that is available comes with a lot of uncertainty; it does not point to exact timestamps where anomalies occurred. Furthermore, the data is high dimensional with over a hundred magnet data collections.

The combination of all of these factors rule out most of the methods covered by Chandola et al. and Pimentel et al. Many methods work only on certain types of datasets (univariate timeseries, continuous values only, etc.), others do not work wel with high dimensional data. There are also methods that rely on a fixed sampling rate to work. Yet other methods rely on precise labels to be trained well. No method has been found in the literature that is applicable to the context of this thesis. This thesis proposes an anomaly detection pipeline built specifically for the MKI magnet data. When it comes to the actual machine learning model that is used in the pipeline, the approach has been developed so that different machine learning models can be plugged into the pipeline and their performance evaluated automatically and compared to other models.

Wells, Ting, and Naiwala state that *"some time series problems can be better served as non-time series problems"* [23]. They analysed the performance of unsupervised learning anomaly detectors for a vehicle related time series problem and showed that a non-time series approach outperformed a time series approach with a significantly improved AUC result. The non-time series approach relies on laws of physics to calculate what the outcome would be at any data point, independent of other data points. It uses equations to calculate the time to impact between a vehicle and an approaching object, and the minimum time required to avoid a crash. Such an approach is not possible in the context of this thesis, as there are too many variables and unknown relationships between the variables to build equations that predict the outcome at any data point.

Some methods rely on dimensionality reduction of data using e.g. PCA and then using the transformed data. This allows for many algorithms which do not work well with high dimensional data to be used. Dimensionality reduction is not explored in this thesis in order to leave room for interpretability of the anomaly detection. An approach which allows for interpretability is preferable so that more insights can be gained about the causes of anomalies. As mentioned in section 3.1, such information could be used to improve designs of LHC components. This is also why popular methods like Deep Neural Networks are not explored.

Feature Selection

Sutha [22] provides a brief review of feature selection algorithms. In her work, Sutha lists different approaches to feature selection, explains how they work, and notes their advantages and drawbacks. She concludes that there is still a need for an effective unified framework for feature selection that is applicable on different types of data and is able to scale up with increasing dimensionality.

Performance Evaluation

When developing any machine learning based application, it is important to use a performance evaluation metric that is suited to the type of problem being solved. A few widely used metrics for evaluation of classification performance have been introduced in 2.4. The choice of one of these was based on the work done by Davis and Goadrich [9], who explored the relationship between PR and ROC curves in the context of highly skewed datasets. Their work is discussed in more depth in subsection 5.4.2.

3.4 Previous Work

This thesis aims to improve on the results achieved by a thesis on the same topic in the academic year 2016–2017 [24]. In his work, Wéry approached the anomaly detection problem using a Gaussian Mixture Model. He devised the first version of feature engineering on CERN's magnet data collections. An improved feature engineering step has been developed for this thesis and is described in subsection 5.1.2.

Next, Wéry trained a GMM on the feature dataset and used it to generate anomaly scores for all data tuples. The hyperparameters for the GMM were chosen based on the Bayesian Information Criterion, which is a statistical criterion developed for model selection. In this thesis, parameters are chosen based on performance in experiments, and a better set of parameters was found than the one suggested by Wéry. This experimental approach is covered in subsection 5.4.3.

He then developed an algorithm that grouped the GMM's anomaly scores into what he named "anomalous segments". This was done to group high anomaly scores that lie near each other in time together so that they can be viewed as one anomaly instead of many anomalies. The amount of anomalies that were grouped was determined by a "top K" parameter; the K worst anomaly scores were taken and then converted into anomalous segments. To evaluate the performance of the GMM, he calculated Precision and Recall using the anomalous segments. Different values for K were tested and an overview of the performance they resulted in was given. Subsection 5.3.1 explains the disadvantages of this evaluation approach and proposes a new one instead. Subsection 5.4.2 introduces a different evaluation metric that uses both Precision and Recall to represent the performance of anomaly detection using one number. This allows for an automated approach towards finding good parameters.

Chapter

4 Data

Before the anomaly detection process can be set up, it is important to get to know the data that will be used for analysis. This chapter introduces and explores the different types of data collections that were made available by CERN. It also introduces a couple of new terms that will be used later on in this text.

4.1 Data Collection Types

Many different measurement collections related to the MKI and the LHC were made available for this thesis. These can be arranged into a handful of types. This section introduces these types.

Tables in this section provide an overview of measurements that belong to a certain type of data collection. They do not, however, list all of the collections with their full names. A list of all data collections can be found in appendix A. Each of the following subsections clarifies the structure of collection names. This is useful to know, as these names are used as labels on figures.

4.1.1 Continuous Data

The first type is fairly straightforward, the Continuous data collections hold measurements on temperatures and pressures inside the MKI magnets. Table 4.1 shows a description of the different measurements that were made. In the temperature measurement descriptions, upstream means that measurements are made at the point where particles enter the magnets, downstream at the point where they leave the magnets.

Measurements for these collections are made with a fixed sampling rate of 2 Hz. However, not all measurements are always stored. CALS (subsection 2.2.3) allows for a filtering strategy that only stores measurements when they change. This strategy is used for Continuous data. Measurements are stored whenever new ones are different from the one stored last, with at most one measurement per second being stored. In case measurements remain constant, fixed interval sampling makes sure that at least one measurement is stored for a certain time interval. This interval is not equal for every collection and could range from fifteen minutes up to a whole day.

Since all changes in Continuous measurements are stored, gaps in collections can be filled using a forward fill approach: they can be filled using the measurements that precede them.

Measurement	Description	Unit	Data type
PRESSURE	Vacuum level in injection kicker magnet	mbar	NUMERIC
TEMP_MAGNET_DOWN	Magnet downstream temperature	°C	NUMERIC
TEMP_MAGNET_UP	Magnet upstream temperature	°C	NUMERIC
TEMP_TUBE_DOWN	Magnet tube downstream temperature	°C	NUMERIC
TEMP_TUBE_UP	Magnet tube upstream temperature	°C	NUMERIC

Table 4.1: Names and descriptions of Continuous data collections.

The measurements in table 4.1 are suffixes of Continuous data collections names. Their full names start with the string "MKI.(A|B|C|D)(5L2.B1|5R8.B2):", where brackets represent different options for substrings separated by vertical bars. The first set of options represents the four magnets (A, B, C, or D), the second one represents the beam that the magnets are for (B1 or B2). An example of a full Continuous data collection name is "MKI.A5L2.B1:PRESSURE".

Continuous data was made available for each of the magnets for both MKI installations. There are 4 magnets, 2 beams, and 5 distinct Continuous measurements. This makes the amount of Continuous data collections equal to 40.

4.1.2 IPOC Data

The second type of data collection is significantly different from the first one. IPOC stands for Internal Post Operational Check, and IPOC data is more closely related to the MKI installations than Continuous data. IPOC data collections consist of voltages and currents measured in the magnets, as well as timing readings. The measurements are described in table 4.2.

IPOC data is measured using an approach called "acquisition trigger sampling": measurements are made based on a non-synchronous trigger which is fired whenever the magnet generators (subsection 2.2.1) pulse. When this trigger fires, all of the IPOC measurements are made simultaneously. A close inspection of the measurement timestamps in subsection 4.3.3 shows that magnet pulses happen at most once every ten seconds. Since these measurements are representative of what is going on inside the magnets, and thus are important for expert analysis, all measurements for IPOC data are always stored.

The values measured for IPOC data change much more frequently and more quickly than Continuous measurements. This, combined with the fact that IPOC data is only measured when the magnets are in use, means that gaps of missing IPOC data cannot be filled using any other values. Also, since the magnets only serve to start LHC runs, which can go on for hours, many large gaps in the IPOC data are to be expected. Gaps like these are clearly visible in figure 5.5.

Names of IPOC data collections start with "MKI.UA(23|87).IPOC.(A|B|C|D)(B1|B2):", where "UA23" only occurs with "B1", and "UA87" only occurs with "B2". Table 2.1

Measurement	Description	Unit	Data type
E_KICK	Energy at injection time	GeV	NUMERIC
I_STRENGTH	Magnet kick strength	kA	NUMERIC
T_DELAY	Kick delay (start point time - trigger time)	µs	NUMERIC
T_FALLTIME	Kick fall time (endThreshold points time - end point time)	µs	NUMERIC
T_LENGTH	Kick length (end point time - start point time)	µs	NUMERIC
T_RISETIME	Kick rise time (start points time - startThreshold point time)	µs	NUMERIC
T_START_TH	Time of startThreshold point	µs	NUMERIC

TABLE 4.2: Names and descriptions of IPOC data collections.

shows that the locations of the MKI installations for beams B1 and B2 are UA 23 and UA 87, respectively.

IPOC data was also available for each of the magnets for both MKI installations. Since there are 7 distinct IPOC measurements, the amount of IPOC data collections is 56.

4.1.3 State Data

State data measures nominal observations; each of the State measurements can be of a predefined set of numbers. The numbers represent specific states that the magnet pulse generator controller (subsection 2.2.1) is in. For example, `MODE` measurements, which measure the mode a pulse generator controller is in, can be "UNKNOWN" (0), "ON" (1), "OFF" (2), or "STANDBY" (3). Tables 4.3 and 4.4 show the distinct State measurements and the values they can take, respectively. The pulse generator controllers mentioned in table 4.3 refer to the systems that control the generators. Remember that one generator powers two magnets, but that there is one controller for both of the pulse generators of an MKI installation (section 2.2).

State data is stored similarly to Continuous data, except that there is no fixed interval sampling. This means that when the measurements stay constant for a long time, gaps in State data collections should be larger than in Continuous data collections. Fortunately, these gaps can also be filled using a forward fill approach.

Measurement	Description	Unit	Data type
CONTROL	Pulse generator controller origin	N/A	TEXTUAL
MODE	Pulse generator controller mode	N/A	TEXTUAL
SOFTSTARTSTATE	Soft-start state	N/A	TEXTUAL
STATUS	Pulse generator controller status	N/A	TEXTUAL

TABLE 4.3: Names and descriptions of State data collections.

State data collections names start with "MKI.UA(23|87).STATE:".

State data is only measured for an MKI installation's pulse generator controller, and State data was available for both MKI installations. Therefore, there are 8 State data collections.

Value	CONTROL	MODE	SOFTSTARTSTATE	STATUS
0	N/A	UNKNOWN	UNKNOWN	UNKNOWN
1	N/A	ON	ON	OK
2	N/A	OFF	OFF	WARNING
3	N/A	STANDBY	BUSY	ERROR
4	N/A	FAULTY	N/A	N/A
5	REMOTE	WARMING_UP	N/A	N/A
6	LOCAL	N/A	N/A	N/A
7	SOFTSTART	N/A	N/A	N/A

Table 4.4: The numerical values that State measurements can take and the states they represent.

4.1.4 Controller Data

Controller data collections hold measurements of configuration parameters that were set for magnet pulses. The parameters determine what a pulse should look like. They set the pulse's amplitude, length, etc. Table 4.5 shows the Controller measurements along with their descriptions. Note that the data types are "VECTOR NUMERIC" rather than "NUMERIC" or "TEXTUAL". This is due to a generic measuring system that was reused at CERN for Controller data. For the MKI installations this thesis focuses on, the first items in the vectors should be used.

Controller measurements are stored in the same way as State measurements, meaning that missing data can be filled using a forward fill approach.

Measurement	Description	Unit	Data type
KICK_COUNT_TOPLAY	Which internal parameters will be used, can be ignored for MKI	N/A	VECTOR NUMERIC
KICK_DELAY_TOPLAY	Requested delay between RF prepulse and trigger	ns	VECTOR NUMERIC
KICK_ENABLE_TOPLAY	Whether or not the magnets are used/pulsed	bool	VECTOR NUMERIC
KICK_LENGTH_TOPLAY	Requested magnet pulse (i.e. kick) length	ns	VECTOR NUMERIC
KICK_STRENGTH_TOPLAY	Requested magnet pulse amplitude	V	VECTOR NUMERIC
KICK_TIME_TOPLAY	Requested delay between beamIn and trigger, can be ignored for MKI	ns	VECTOR NUMERIC

Table 4.5: Names and descriptions of Controller data collections.

Controller data collection names start with "MKI.UA(23|87).F3.CONTROLLER:".

Like State data, Controller data is only measured for an MKI installation as a whole. Collections for both MKI installations were made available. Because there are 6 distinct Controller measurement per MKI installation, there are 12 Controller data collections.

4.1.5 LHC Data

Two types of measurements remain, they are listed in table 4.6. These measurements measure LHC beam parameters, instead of an aspect of the injection kicker magnets. `BEAM_INTENSITY` measures the total charge of a particle beam moving in the LHC, while `BUNCH_LENGTH_MEAN` measures the mean length of the bunches in the beam (subsection 2.1.2).

These two measurements are in a direct relationship with the amount of particles in the LHC. Whenever a new experiment is started, the values for these measurements will start growing for a certain amount of time, after which they decay while the experiment goes on. When the experiment ends, the values drop back down to 0, as the particles are then ejected from the LHC. While an experiment is being started, `BEAM_INTENSITY` will grow more than `BUNCH_LENGTH_MEAN`, because it measures the total charge of the whole beam, whereas the length of the bunches in the beam should be constant.

If the values for these measurements are larger than 0, that means that the MKI magnets must have been used (to inject particles into the LHC). The opposite is not necessarily true, however: it is possible for these measurements to be 0 after the magnets have been used for a period of time. That can happen during e.g. the KISS process, testing procedures, etc.

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

TABLE 4.6: Names and descriptions of LHC data collections.

Names of `BEAM_INTENSITY` collections start with "LHC.BCTFR.A6R4.(B1|B2):", while the ones for `BUNCH_LENGTH_MEAN` start with "LHC.BQM.(B1|B2):".

Since LHC data is measured for both beams, there are 4 LHC data collections. This brings the total amount of all of the data measurement collections to 120.

4.1.6 Electronic Logbook Data

Lastly, a collection completely different from the other ones has been provided by CERN. This collection, "MKI.ELOGBOOK_tagged", contains manually created logbook entries. The entries are created at CERN for several reasons. For example, entries can denote that an inspection has been performed, or they can simply provide some information that is useful to other employees at CERN. More importantly, logbook entries can be created to denote that an anomaly has occurred in the machines.

Electronic logbook entries have been grouped into several types. They are:

- **anomaly:** Denotes that equipment was faulty and that an anomaly should be detected.
- **fault:** Equipment was faulty, but the fault was either insignificant or undetectable because there were no sensors that could detect the fault.

- **info:** Information useful for others at CERN.
- **intervention:** Used for equipment intervention. Possible interventions are hardware exchanges, software changes, or parameter changes.
- **research:** Equipment or beam parameters are tested.

Note that both interventions and research can impact measurements. From the point of view of the data, whenever interventions happen or research is done, the resulting behaviour of this in the data can also be classified as anomalous. The difference from anomalies is that is is known at CERN when these events occur. Therefore, it would be interesting to see how well an anomaly detector performs when trying to detect anomalies defined as above versus anomalies defined as any of the definitions for "anomaly", "intervention", or "research". Subsection 5.3.3 explores this idea further.

Because the electronic logbook entries play an important role as labels in the evaluation of the anomaly detection process, they will hereinafter be referred to as labels.

Definition 4.1.1. An **anomaly label** is a label that is of type "anomaly".

Analogous definitions hold for the other types of labels.

Label Attributes

Table 4.7 shows the most important attributes of an anomaly label. It shows that the type of the label can by identified using its "TAG" attribute. The table also shows that each label is associated to one of the MKI installations. Remember that "MKI2" is the installation that is used for beam 1 (table 2.1). This means that each of the labels is relevant for only one beam, and thus for only one MKI installation.

Appendix B contains more labels of different types.

EVENTDATE	25/07/2016 00:19:48
PATH	LHC.MKI2
COMMENT	CCC calls for a faulty MKI2 generator during LHC fill. I checked and there was a flashover in magnet D, vacuum recovered rather slowly. IPOC also shows an increased current for IM-D. Flashover **not** detected by fast interlocks. ...
TAG	anomaly

Table 4.7: An anomaly label.

Every label contains an "EVENTDATE" attribute which indicates the timestamp on which the event described in the label occurred. Note, however, that for anomaly labels, the "EVENTDATE" is only an approximation of when the event occurred, as its exact moment cannot be determined due to the complex behaviour of anomalies. Often, the timestamp is just set to the moment the label was entered into the

logbook. Expert knowledge at CERN instructs that the anomaly could have occurred anywhere up to 12 hours before the timestamp.

Furthermore, anomaly labels can only be created after anomalies have been detected by CERN, and it is possible that some anomalies have been overlooked. This can happen when anomalous behaviour was not severe enough to stop the pulse generators (thanks to the BIS mentioned in section 2.1) or to have an impact on the particle beams.

4.2 Beams B1 and B2

There are slight differences in the IPOC measurement timestamps between the two MKI installations. Within one beam, all IPOC measurements are made at the exact same timestamp thanks to CERN's acquisition trigger sampling system. However, the timestamps are not the same across both beams B1 and B2, because the two MKI installations are operated asynchronously. Table 4.8 demonstrates this fact for a few `E_KICK` measurements. NaN, not a number, is used to represent that there is no data in a collection for the given timestamp. Because the data is measured at different moments between the two beams, data for one beam is missing when data for the other one is present. Looking at the timestamps, it is also apparent that they cannot be shifted to align the data for the two beams, as the time periods between successive measurements are different between the two beams in this example.

Timestamp	**MKI.UA23.IPOC.AB1:E_KICK**	**MKI.UA87.IPOC.AB2:E_KICK**
2016-09-12 03:46:33.788	NaN	401.04
2016-09-12 03:46:43.788	NaN	401.04
2016-09-12 03:48:33.086	450.00	NaN
2016-09-12 03:49:13.887	NaN	450.00
2016-09-12 03:55:31.887	450.00	NaN
2016-09-12 03:56:12.686	NaN	450.00

TABLE 4.8: `E_KICK` measurements for magnets A for beams B1 and B2.

In the anomaly detection process, it is preferred to have as much data as possible so that the anomaly detection model can be trained as well as possible. As discussed in subsection 4.1.2, gaps in IPOC data cannot be filled with any other data. Fortunately, the gaps in all other collections can be filled using a forward fill approach, such that there is a measurement in every collection for every IPOC measurement timestamp. The IPOC data collections for B1 and B2 cannot be joined together, however, because of the fact that IPOC data gaps cannot be filled.

4.2.1 Problem Statement Adaptation

The fact that all of the data collections can be joined within one beam, but not across both beams, has an important consequence: the anomaly detection process cannot be applied to datasets that consist of data for both beams. Because of this, the

anomaly detection problem has been reshaped: the data will be split into two groups, one group for each beam, and anomaly detection will be applied on both groups separately; evaluation will happen for both groups separately as well. Remember that each label is associated with one beam (subsection 4.1.6), which makes separate evaluation possible.

4.3 Data Exploration

This section displays a few of the data collections introduced in the previous section so that an idea can be given of what the data looks like. For this thesis, data was available between 2015-06-22 and 2016-09-14. There was a break around the end of 2015, though. This is the year-end technical stop, in which maintenance and tests are performed.

4.3.1 Continuous Data

Figure 4.1 shows pressure measurements for both beams B1 and B2 throughout 2015 and 2016. The pressure is considerably lower overall in 2016. Also, it shows similar behaviour across both beams in time, which makes sense. For normal LHC runs, both beams will be used, which will cause the pressure in both MKI installations to grow at the same time.

Figure 4.2 shows `TEMP_MAGNET_UP` for both beams in both years. These measurements vary to a much lesser extent than the pressure measurements. However, the measurements are considerably higher in 2016 in general, instead of lower as was the case for pressure. This is also the case for all other temperature measurements (not shown here). Temperatures show similar behaviour across both beams in time as well. The LHC cycles are more visible in the temperatures. The temperatures increase each time particles are injected into the beam, and then decrease as the experiments finish and/or particles are ejected from the LHC.

The periods where the measurements remain relatively stagnant are period where no LHC runs were done. The values do still show some activity due to testing and development procedures being executed.

4.3.2 IPOC Data

Figure 4.3 displays the `I_STRENGTH` for both beams in 2016. A pattern is clearly visible here. All magnets seem to be used with strengths that grow in steps. This can be attributed to the KISS conditioning procedure briefly explained in subsection 2.2.2.

Figure 4.4 shows the `T_RISETIME` for both beams in 2015. It shows that the values very consistently stay in the range [654 µs, 670 µs]. The risetime does appear to lie outside this range at moments. This could have happened due to anomalies.

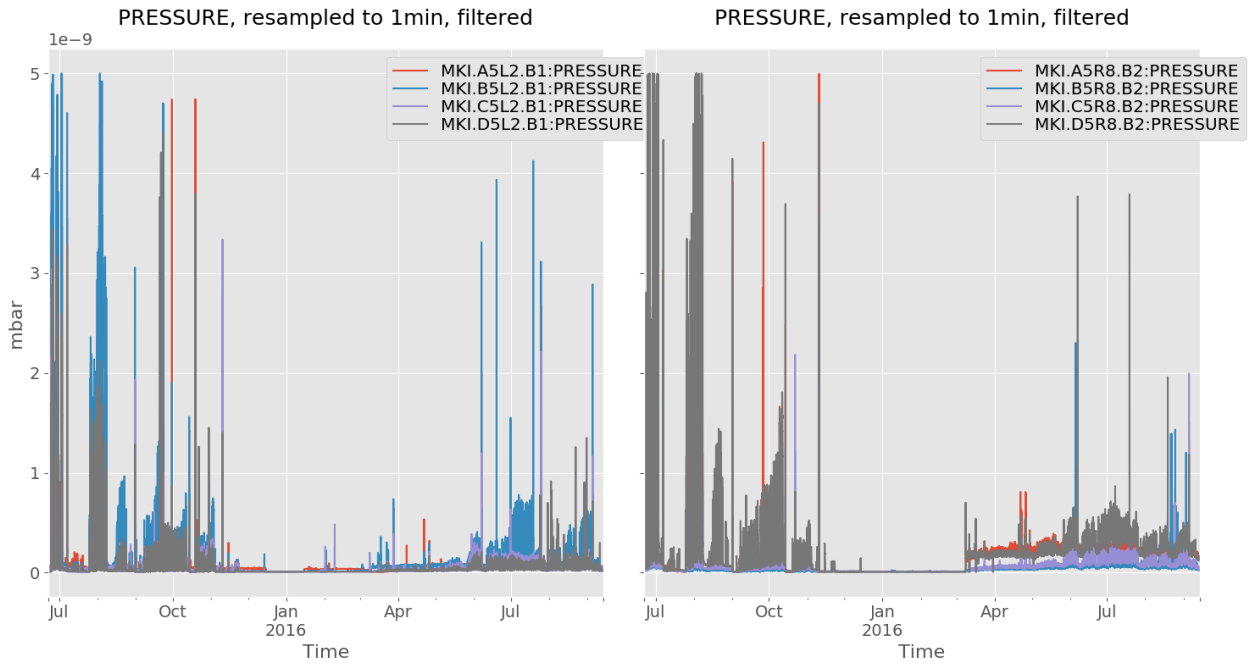


FIGURE 4.1: `PRESSURE` measurements for both beams.

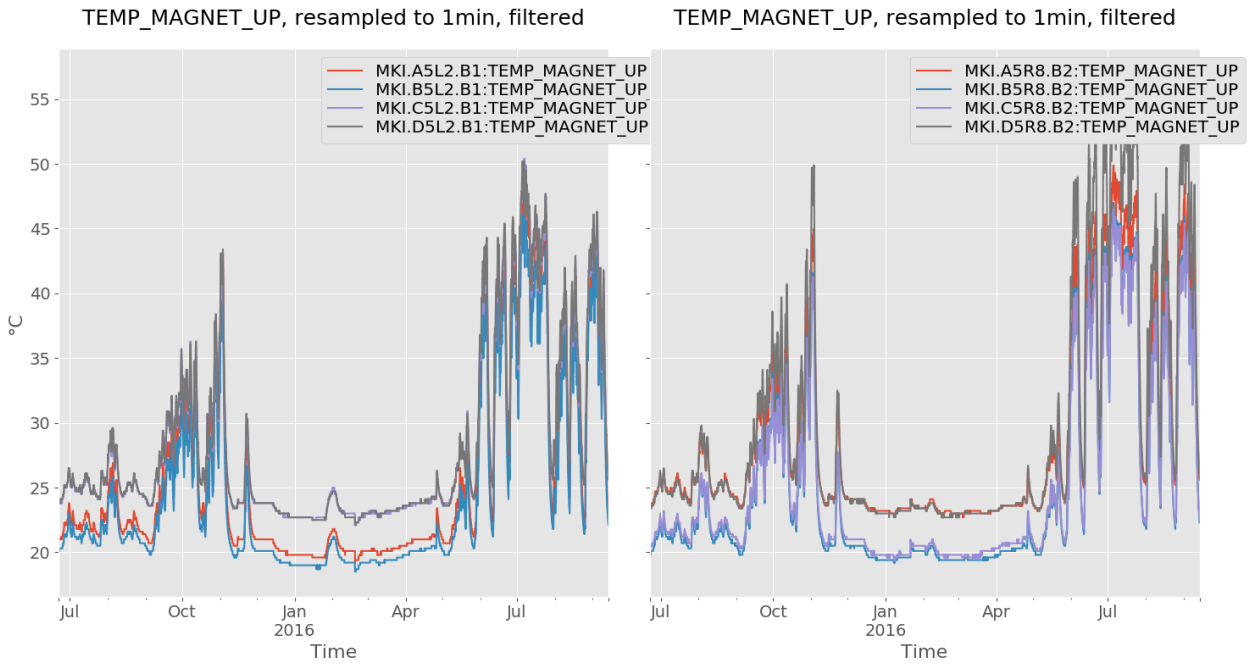


FIGURE 4.2: `TEMP_MAGNET_UP` measurements for both beams.

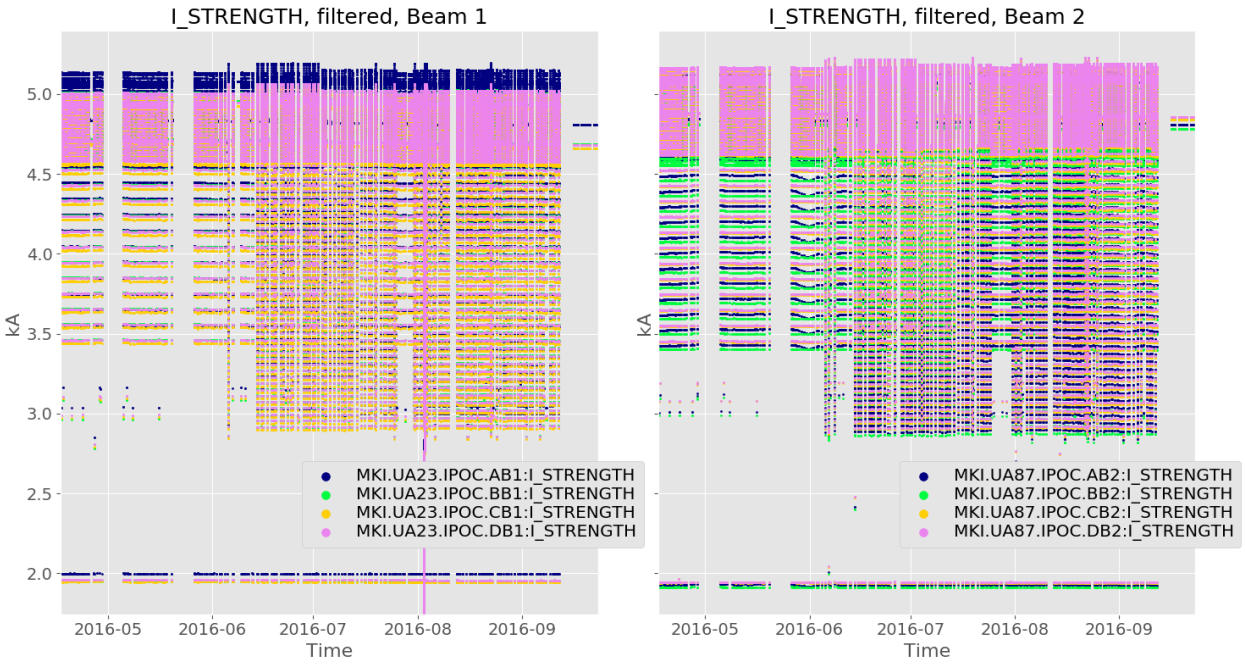


Figure 4.3: `I_STRENGTH` measurements for both beams in 2016.

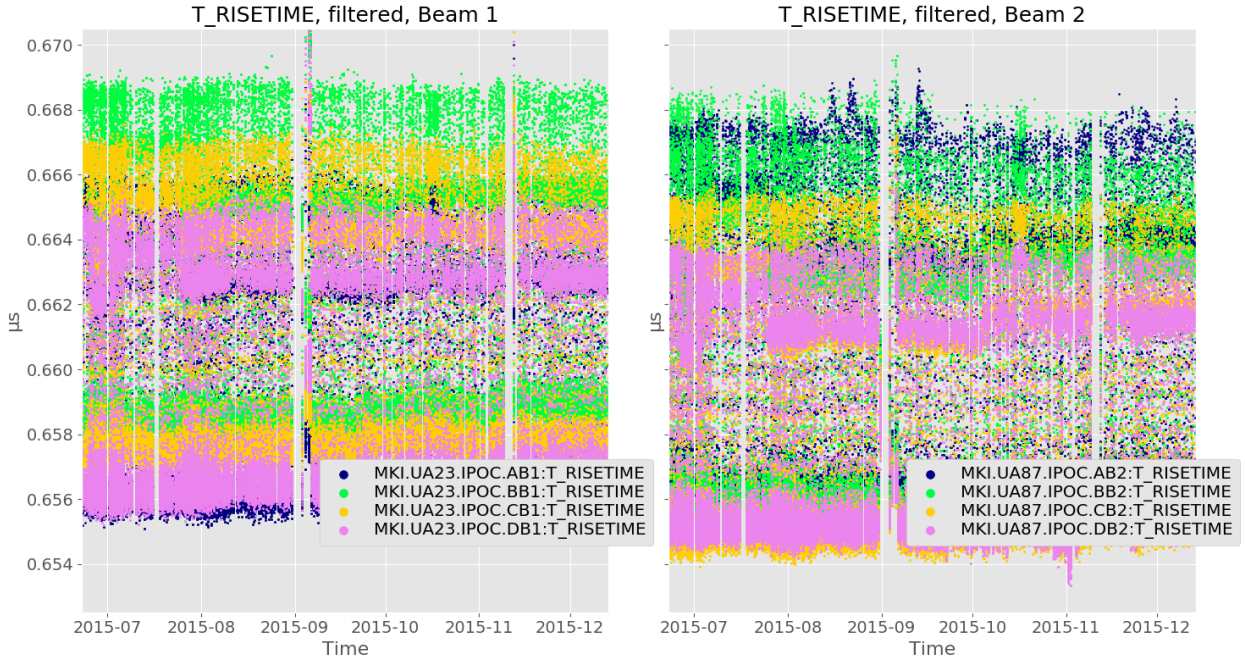


Figure 4.4: `I_RISETIME` measurements for both beams in 2015.

4.3.3 IPOC Data Timestamps

Figure 4.5 shows the distances between successive IPOC measurement timestamps for a few months in 2015. There is a pattern in the distances, but is hard to see thanks to the few very large distances that skew the image. These larger distances are the result of the magnets not being used for longer periods of time, i.e. a few days.

Figure 4.6 shows a zoomed in view of figure 4.5 and shows the patterns that were previously invisible. Remember that IPOC measurements are made when the MKI magnets pulse. The bands of distances at low values in the figure show that many measurements are made at specific time periods. This means that the magnets often pulse at specific frequencies, although they also pulse irregularly as shown by the rest of the distances.

Another interesting fact is shown in figure 4.7: the smallest distance between two IPOC measurement timestamps is 10 seconds, or 9.9 to be exact. That means that magnet pulses happen at most once every 10 seconds. These pulses can be attributed to the KISS process described in subsection 2.2.2. From "KISS conditioning of the LHC injection kickers" [1]: *"The trigger signals that pulses the MKI during the "KISS" are provided by a local pulse generator setup. It's interval can be regulated from 1.5 s up to 9.9 s. Typically the pulse interval is set to 9.9 s."*

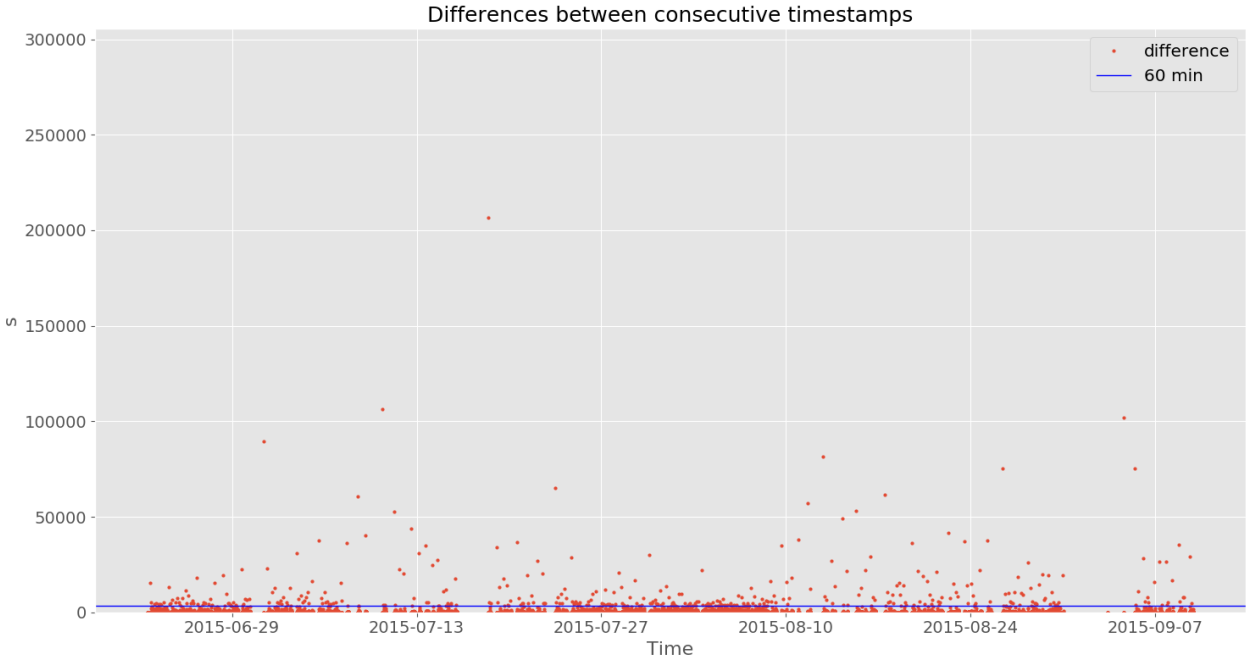


FIGURE 4.5: Differences between successive IPOC measurement timestamps in seconds.

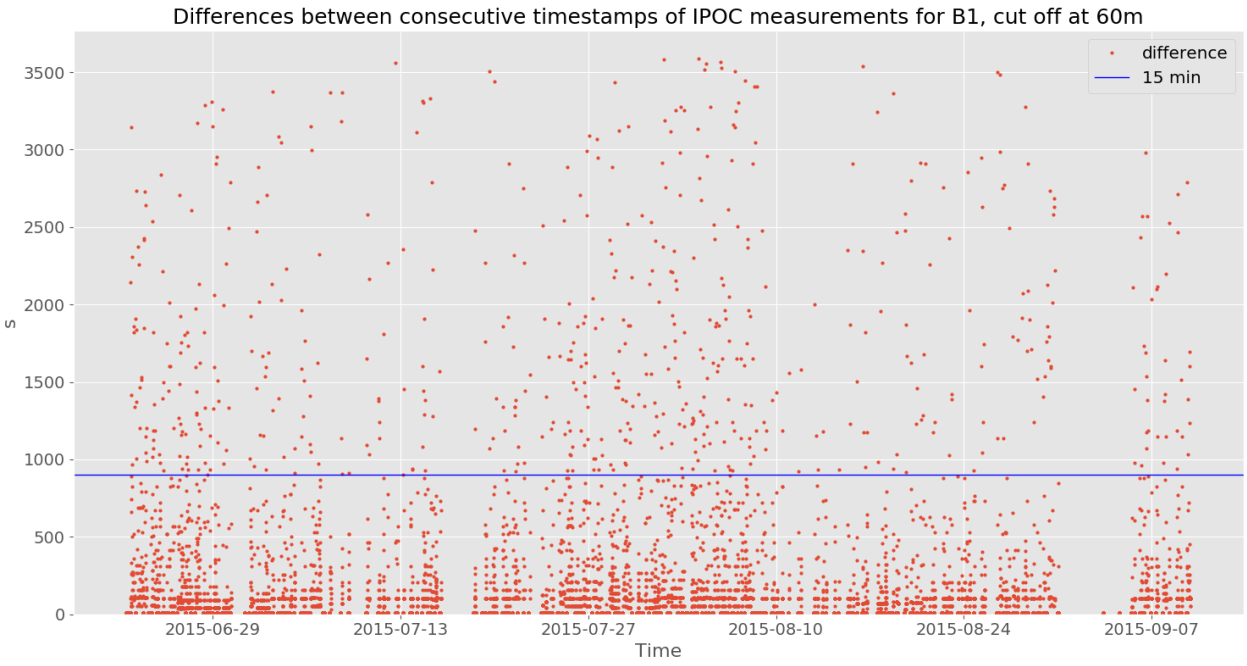


Figure 4.6: Differences between successive IPOC measurement timestamps in seconds, cut off at 60 minutes.

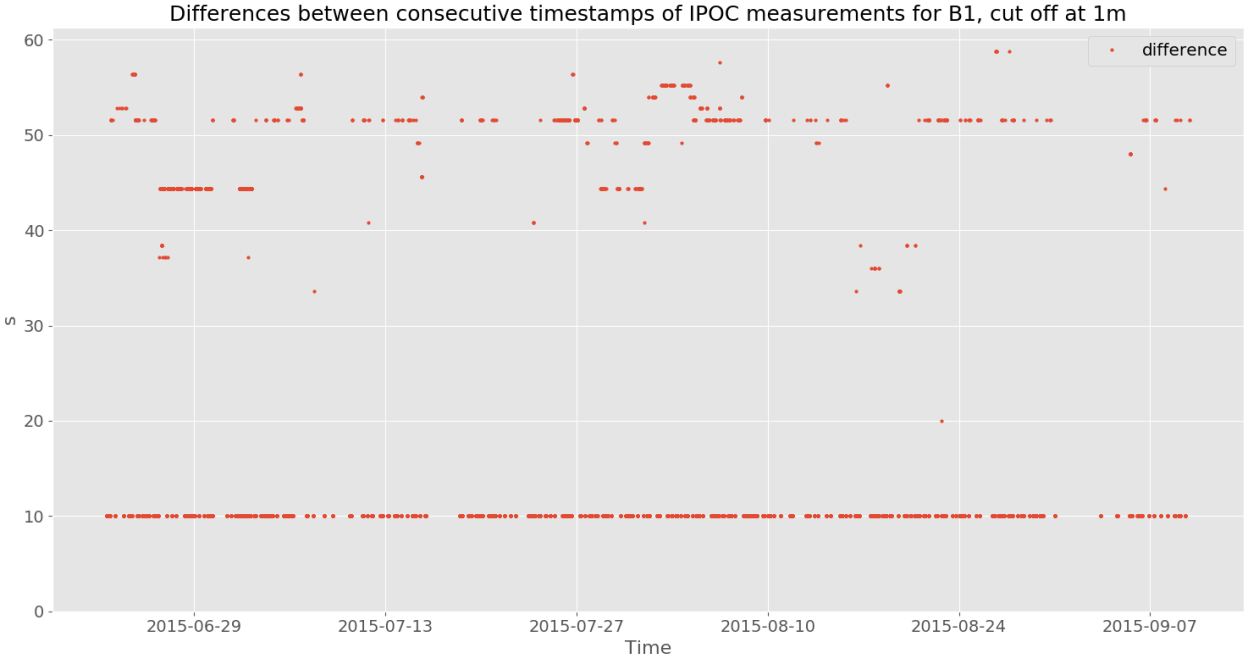


Figure 4.7: Differences between successive IPOC measurement timestamps in seconds, cut off at 1 minute.

4.3.4 LHC Data

`BEAM_INTENSITY` changes very often, as can be seen in figure 4.8. It grows each time an experiment is started. It then decreases as experiments go on and particles collide. When the experiment finishes, the particles are ejected from the LHC and `BEAM_INTENSITY` quickly drops back down to 0.

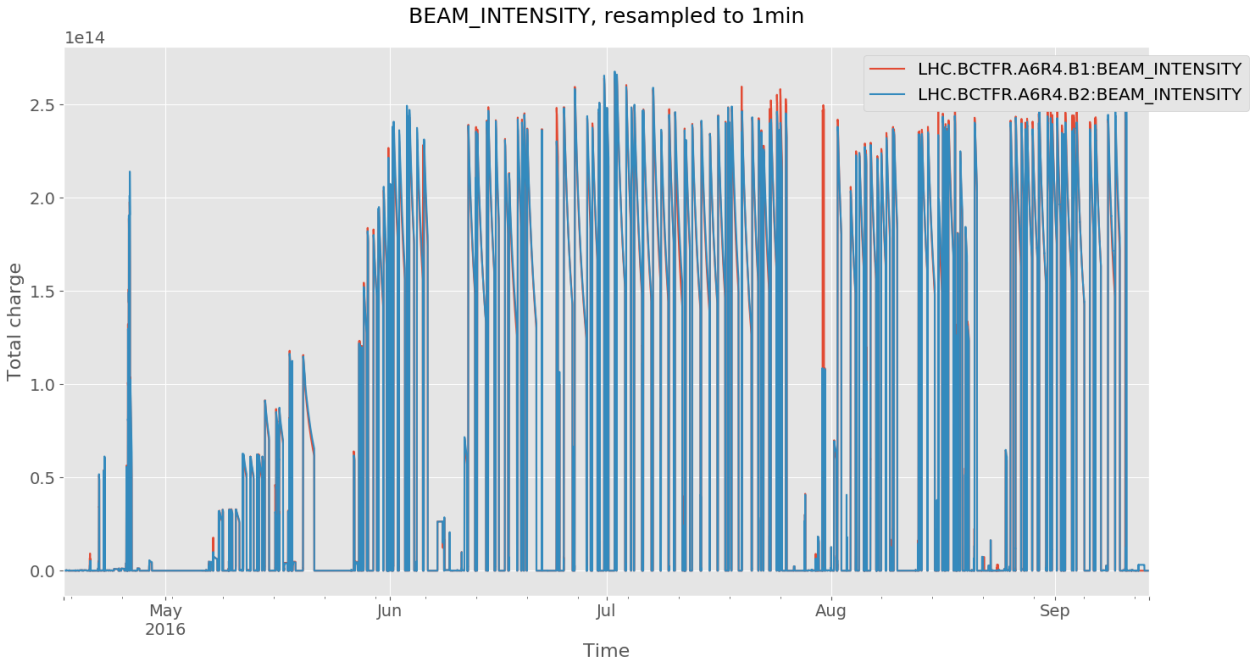


FIGURE 4.8: `BEAM_INTENSITY` for both beams in 2016.

4.3.5 Labels

Table 4.9 shows the amounts of different types of labels (e-logbook entries) that were created by CERN. Where the anomaly labels lie in time can be seen in chapter 6.

Label type	**Beam 1**	**Beam 2**
anomaly	23	24
fault	11	34
info	75	134
intervention	33	62
research	10	20
Total:	152	274

TABLE 4.9: Amounts of different types of labels.

4.4 IPOC Segments

As described in subsection 2.3.4, each data tuple will receive an anomaly score in the anomaly detection process. After the assignment of anomaly scores, the scores will be evaluated so that a statement can be made about the quality of the process. Evaluating the correctness of those scores is not a trivial task, however.

Since labels do not point to an exact timestamp, anomaly scores cannot be evaluated at each of the timestamps individually. CERN has advised that the actual anomaly could lie anywhere up to 12 hours before the timestamp of a label. This uncertainty calls for an unconventional evaluation approach (which is described in section 5.3). "IPOC segments" have been introduced to mitigate the uncertainty of the labels.

Definition 4.4.1. An **IPOC segment** is a set of data for a period of time during which the MKI magnets were used to fulfil a certain goal. Such a goal can be the initiation of an experiment, a testing procedure, or any other goal.

IPOC segments get their name thanks to the IPOC data collections. Since IPOC data is only measured when the magnets are in use, these collections will be used to find periods for which the magnets were active. There is a problem involved in finding these period, though. It is not exactly clear when segments start and when they end. The time periods for which to create segments will be determined using a parameter called `segmentation distance`. The process of transforming data sets into sets of segments will be called "segmentation".

Since the term "IPOC segments" will be used a lot throughout the remainder of this text, they will be simply referred to as "segments".

4.4.1 Segmentation

Definition 4.4.2. The parameter **`segmentation distance`** is the distance between successive measurement timestamps that decides whether or not the measurements belong to the same IPOC segment. Whenever the distance between successive timestamps exceeds the `segmentation distance`, the measurements are said to belong to two different segments.

Different values for `segmentation distance` will cause data sets to be transformed into different sets of segments. Example 4.4.1 shows the effect of different values on the segmentation. Another way to illustrate the parameter is as a threshold on the distances displayed in figure 4.5. At every timestamp where the distance exceeds the parameter, a new segment is started.

Example 4.4.1. Table 4.10 shows the segmentation of a set of timestamps with segmentation distances of 15 and 30 minutes. For 15 minutes, the segmentation process creates three segments. Alternatively, two segments are created for the distance of 30 minutes.

Timestamp	Distance	Seg.nr. if d = 15 min	Seg.nr. if d = 30 min
00:00:00	N/A	1	1
00:01:00	1 min	1	1
00:02:00	1 min	1	1
00:20:00	18 min	2	1
00:21:00	1 min	2	1
00:22:00	1 min	2	1
00:53:00	31 min	3	2
00:54:00	1 min	3	2

TABLE 4.10: The segmentation process with `segmentation distances` of 15 minutes and 30 minutes.

The `segmentation distance` parameter has a large impact on the evaluation of anomaly scores. If it is too large, segments will consist of more than one machine usage period. If it is too small, too many incomplete segments will be created. Subsection 5.4.3 describes how different values for the parameter can be explored so that a well performing one can be chosen.

4.4.2 Anomaly Score

Just as a data tuple can be assigned an anomaly score, an IPOC segment can be assigned one as well. The anomaly score of a segment can be calculated using the anomaly scores of the data tuples that it consists of. Many different approaches towards the calculation of a segment's anomaly score are possible, section 5.3 introduces a few of them.

Definition 4.4.3. An **anomalous segment** is a segment that contains anomalous behaviour. A segment can be marked as anomalous if its time period falls in the 12 hour time window of an anomaly label.

4.4.3 Problem Statement Adaptation

At this point, the anomaly detection problem can again be respecified slightly: instead of trying to find anomalous behaviour that happens at any specific moment, the goal will be to detect anomalous segments.

4.4.4 Segment Length

Different values for `segmentation distance` result in different sets of IPOC segments. Figures 4.9 and 4.10 display distributions[1] of lengths of segments for varying `segmentation distance` settings. Figure 4.9 displays distributions for beam 1, figure 4.10 for beam 2.

[1]The distributions were approximated by Kernel Density Estimation using a Gaussian basis function and kernel bandwidth of 3.

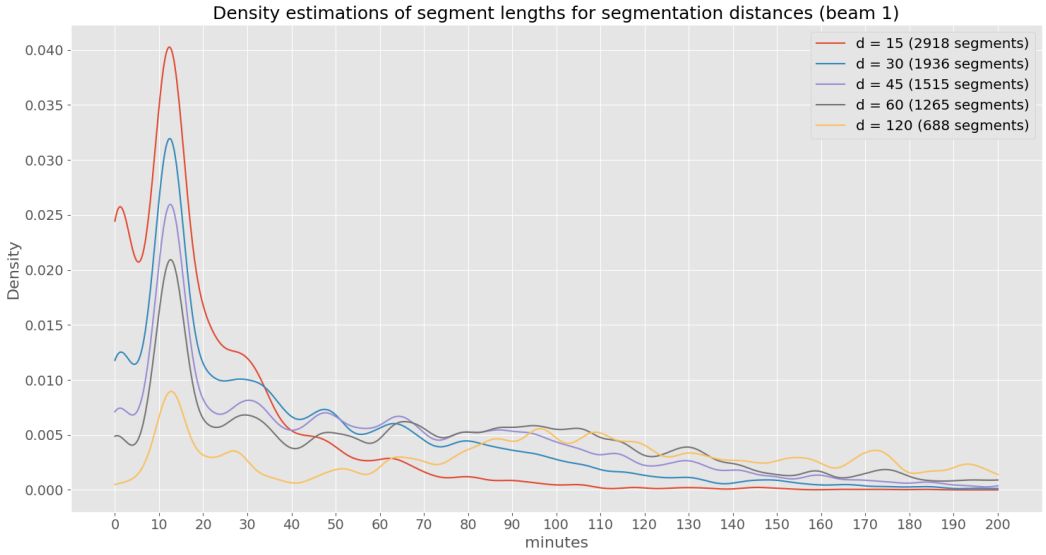


Figure 4.9: Distributions of lengths of beam 1 IPOC segments for different segmentation distances.

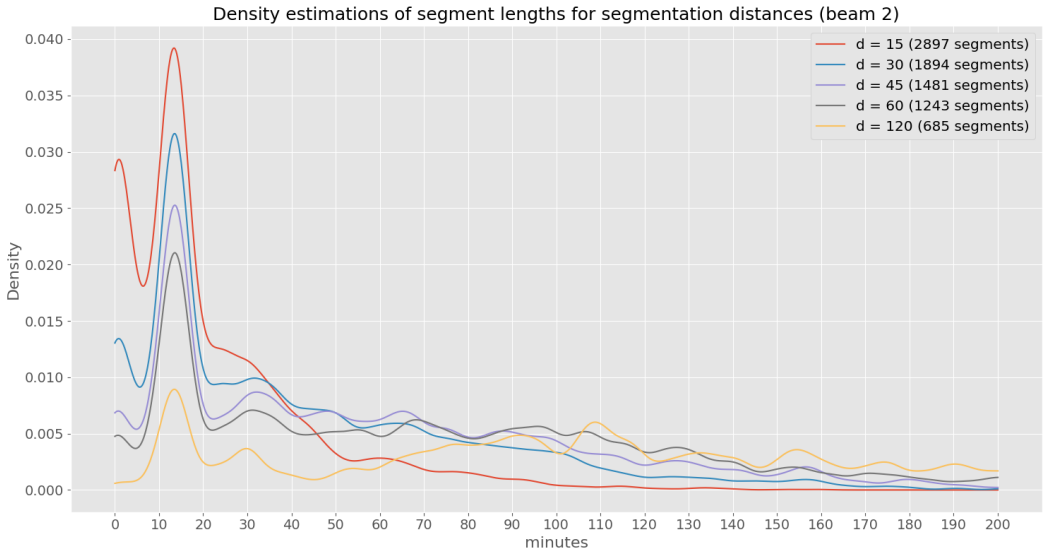


Figure 4.10: Distributions of lengths of beam 2 IPOC segments for different segmentation distances.

In the legends of the figures, the resulting amount of segments for each distance is shown as well. Note that as the `segmentation distance` increases, the amount of resulting segments decreases. This is to be expected, as larger distances will allow for segments to consist of more points, which leads to a lower amount of segments to be created. Distances of 120 minutes or higher should not be used, as at that point, a lot of segments will contain multiple sessions of magnet usage. This can be verified with figures 5.5 and 5.6, which show a zoomed in view of IPOC measurements.

There is a clear pattern visible in both figures. The distributions are at their highest points for lengths between 10 and 20 minutes, regardless of the distance. In other words, most of the segments last for less than 20 minutes. Noteworthy is that for both beams, a second peak has grown between lengths of 90 and 120 minutes for a `segmentation distance` of 120 minutes. This is due to the same phenomenon as described above; the segments start containing multiple sessions of machine usage. The peaks near 0 can be attributed to the magnets sometimes being pulsed for testing purposes.

It appears that distances between 30 and 60 minutes are good options, as the distributions for 30, 45, and 60 minutes are very similar. The distributions for 15 and 120 minutes are significantly different from the other three.

The distributions look very similar across both figures. That means that the magnets were used for approximately the same time periods in both beams. The slight variations between the figures can be attributed to the particle beams not being exactly the same when they are being injected and to testing procedures where only one MKI installation is used at a time.

Chapter

5 Anomaly Detection Pipeline

This chapter describes in detail the anomaly detection process that has been developed. The data examined in chapter 4 will flow through four distinct steps during this process. First, preprocessing is applied on the data to transform it into a feature set ready for anomaly detection. In the second step, an anomaly detector is trained using the data in the feature set and anomaly scores are generated for each data tuple. Third, the output of the second step is transformed into a set of IPOC segments (section 4.4) with anomaly scores. The scores of these segments are then used to determine whether or not they are anomalous, while labels (subsection 4.1.6) are used to determine which of the segments are truly anomalous. Finally, the quality of the whole process is evaluated in the evaluation step.

This process is called the anomaly detection pipeline by analogy to a physical pipeline. The data flows in one direction through all of the steps, each step relying on the output of the step before it. In a physical pipeline, each of the components can be replaced for a different one that does the same job. Similarly, many approaches to the different steps in the anomaly detection pipeline are possible, as long as the purpose of the step is fulfilled. This chapter presents multiple options for some of the components of the pipeline. Subsection 5.4.3 explains how alternative approaches can be used and evaluated.

5.1 Preprocessing

5.1.1 Data Filtering

In order to make sure that the anomaly detection model can be trained correctly, the data needs to represent well what the actual situations were in the LHC. It turns out that the data prepared for this thesis contains erroneous measurements. Examples of this are temperatures and pressures which lie orders of magnitude higher than their means, or timing measurements less than zero. Negative timing measurements are impossible, of course, but temperatures and pressures in the magnets outside certain ranges are said to be impossible in as well by CERN. Table 5.1 shows the ranges, determined by CERN, in which measurements must lie. Measurements that lie outside the range are to be filtered out of the data.

Measurement	Minimum	Maximum
PRESSURE	9×10^{-12} mbar	5×10^{-9} mbar
TEMP_MAGNET_(DOWN|UP)	18 °C	60 °C
TEMP_TUBE_(DOWN|UP)	18 °C	120 °C
I_STRENGTH	1 kA	N/A
T_DELAY	10 µs	N/A

Table 5.1: Ranges in which certain measurements can be assumed to be valid. The ranges have been determined by expert knowledge at CERN.

Data can be filtered in two ways: either each erroneous measurement can be deleted individually, or the whole data tuple at the timestamp of the measurement can be deleted. Both of these approaches are used in the filtering step. When Continuous measurements are erroneous, they are deleted individually, as they can simply be replaced using the fill forward approach mentioned in subsection 4.1.1. On the other hand, erroneous IPOC measurements cannot be deleted individually to then be replaced by their preceding measurements (subsection 4.1.2). For this reason, when one of the IPOC measurements in table 5.1 is erroneous, the whole data tuple at the timestamp of the measurement is deleted.

The figures in subsection 4.3 show data that has already been filtered. This was done so that erroneous measurements would not skew the plots' data ranges. The following figures illustrate what the raw data looks like before and after filtering for a couple of data sets. For the data collections that are not included in the figures, similar statements can be made as the ones throughout the rest of this subsection.

Figure 5.1 displays the worst effect of erroneous measurements that can be encountered in the data. The temperatures for magnets A and D in beam 2 are so high that they tower far above all of the other data and completely skew the data distributions. If these measurements were used, an anomaly detector would surely assign higher anomaly scores to the data tuples they lie in. Figure 5.2 shows a different set of temperature measurements. In this case, there are no temperatures of up to 3000 °C that drown out the shape of the raw data. There's only a small set of temperatures that are deleted early May.

Note that in figures 5.1 and 5.2 there is never a period of time in which the data remains constant at one of the ends of a filter range. This confirms that the filters don't remove a period of data for which the measurements might have truly lied outside of the filter range; there is no cut-off of relevant data.

Figure 5.3 shows `T_DELAY` measurements. In the raw data, there are 4 bands in which most of the measurements lie, with two of the bands lying around 0 µs and −15 µs for both beams. These two bands are clearly made up of erroneous measurements, as the time delay for `T_DELAY` cannot be zero or less. The measurements are removed thanks to the `T_DELAY` minimum value of 10 µs. Peculiarly, the other measurements that do not fall in one of the two remaining bands are also deleted, even though they are acceptable according to the filter. They are deleted due to the fact that all IPOC measurements are made simultaneously and that whole data tuples

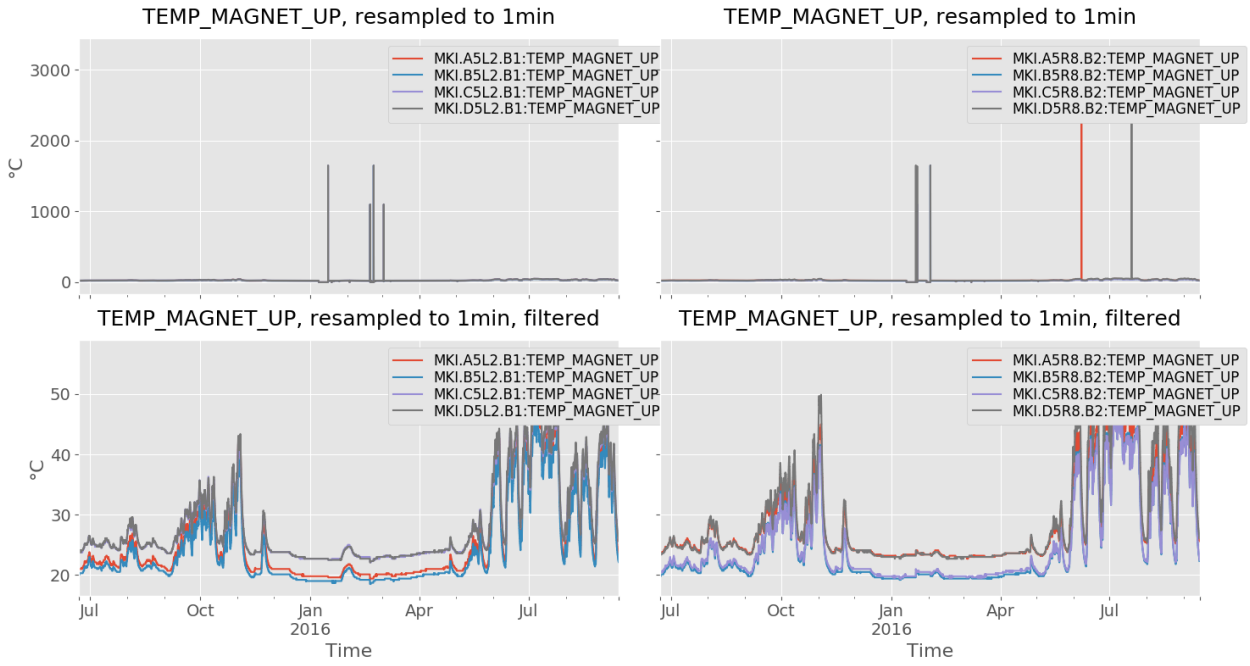


FIGURE 5.1: `TEMP_MAGNET_UP` measurements before (top) and after (bottom) filtering.

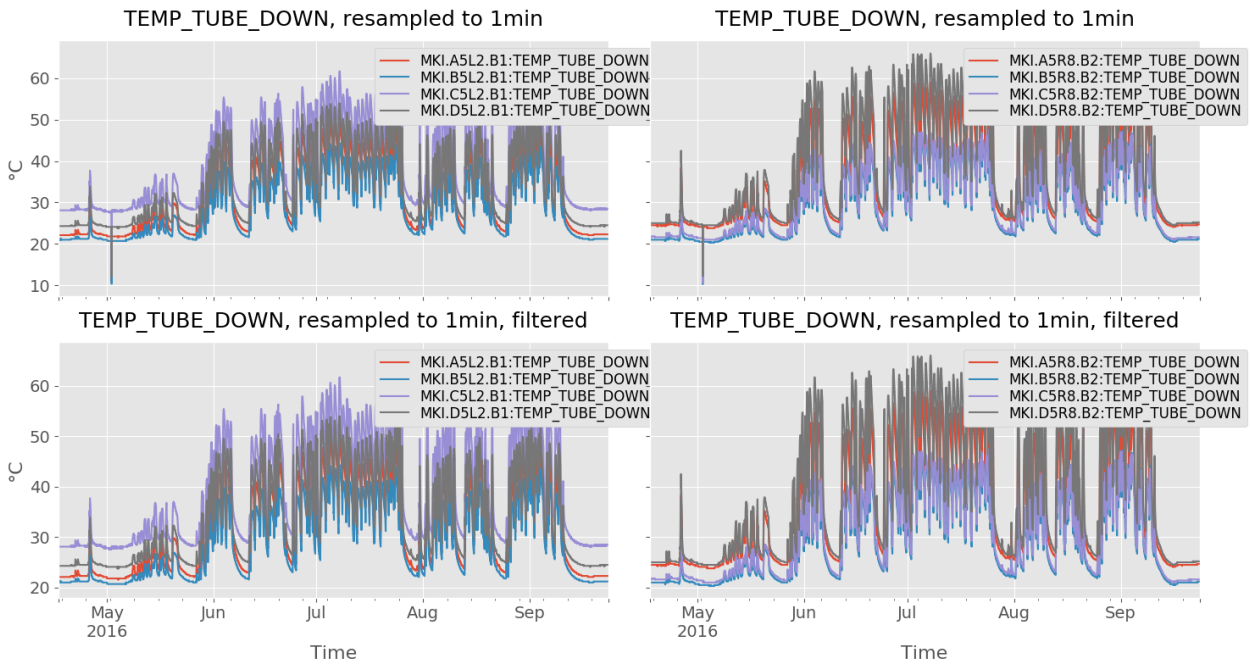


FIGURE 5.2: `TEMP_TUBE_DOWN` measurements before (top) and after (bottom) filtering.

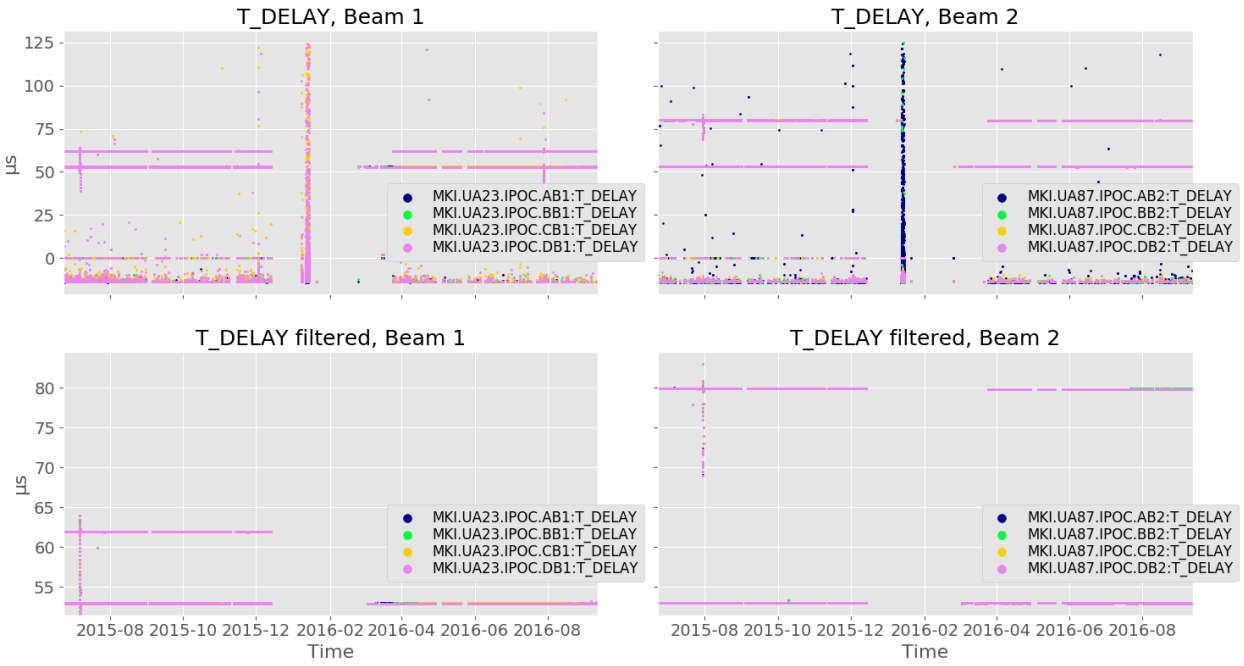


Figure 5.3: `T_DELAY` measurements before (top) and after (bottom) filtering.

Figure 5.4: `T_RISETIME` measurements before (top) and after (bottom) filtering.

are deleted for erroneous IPOC measurements. When there is an erroneous IPOC measurement for one of the magnets, the other ones tend to be erroneous as well. This makes the removal of whole data tuples when erroneous IPOC measurements occur a good choice. Furthermore, CERN has confirmed that when there is an erroneous IPOC measurement, the other ones should be regarded as erroneous as well. Note that the vertical band of `T_DELAY` measurements at the end of 2015 falls in the period of the LHC's year-end technical stop and can be attributed to testing procedures.

The effect of removing whole data tuples is not only useful for `T_DELAY` collections, but also for all of the IPOC collections. Figure 5.4 shows `T_RISETIME` measurements, which mostly lie around the same band for both beams. Note that table 5.1 does not hold any filter values for `T_RISETIME`, but that erroneous measurements are deleted anyway. These are caught thanks to the filters for `I_STRENGTH` and `T_DELAY`. This means that whenever IPOC collections are filtered, `I_STRENGTH` and `T_DELAY` measurements should be present in the data tuples as well so that erroneous measurements in other collections will be deleted correctly.

Data shown throughout the rest of this text can be assumed to be filtered data, unless explicitly stated otherwise.

5.1.2 Feature Engineering

Aside from ensuring an anomaly detector learns from correct data, it is important to transform the data collections into one dataset that can be passed to the detector. Such a dataset should consist of features that are relevant for identifying anomalies. Generally, the presence of irrelevant features reduces the detector's learning performance and computational efficiency [15]. Feature engineering is the process of selecting or extracting those features that are relevant.

IPOC Data

Since the IPOC measurements are most representative of what is going on inside the MKI magnets, they will form the core of the feature set. However, this decision severely limits the amount of data that can be used in the rest of the pipeline, since gaps in IPOC data cannot be filled (subsection 4.1.2). The data has shown that IPOC measurements occur at most once every 10 seconds. A closer inspection of the IPOC data shows that there are relatively large gaps after most sessions of magnet usage. Figures 5.5 and 5.6 show closer views of IPOC data collections of magnet A of beam B1. The data has been normalized in order to make it easier to see when the measurements took place rather than what their real values are. The existence of the IPOC segments defined in section 4.4 is clearly visible in these figures. The large gaps after IPOC segments are also visible. These gaps are expected to be in the IPOC data, because LHC experiments run for hours and the magnets are only used to start those experiments (by injecting particle beams). This makes the IPOC measurements a good choice as the core of the feature set. By using data that was measured at the timestamps for which IPOC measurements exist, the feature

dataset is guaranteed to only contain data that is relevant for the anomaly detection problem at hand. Data at other timestamps can be discarded, since it was measured when the magnets were not in use. After all, the goal is to detect anomalies in the behaviour of the magnets, which makes data measured when the magnets were not in use irrelevant[1].

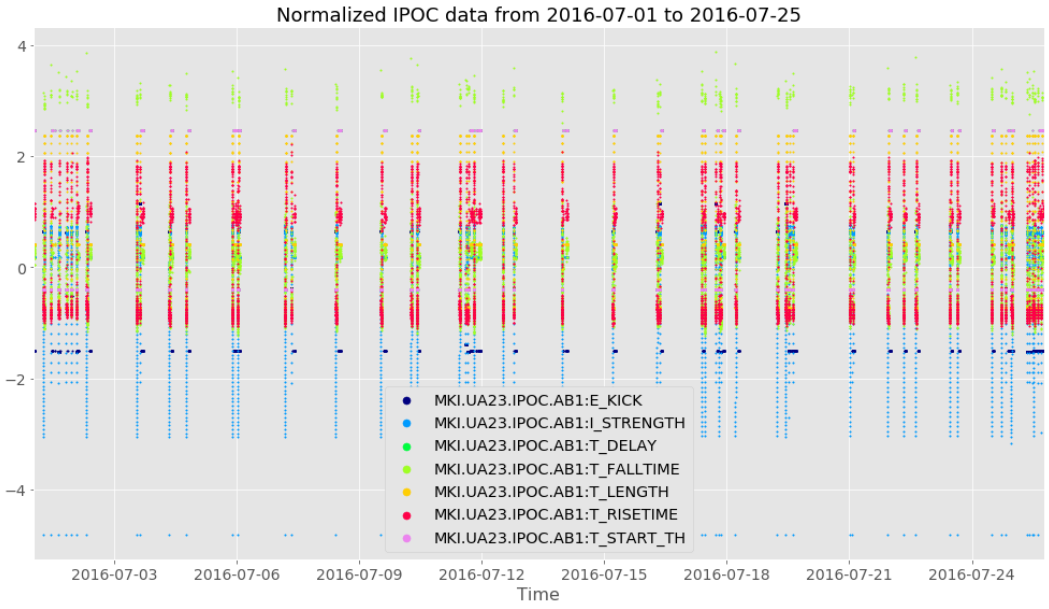


Figure 5.5: A closer view of (normalized) IPOC data for July of 2016.

Non-IPOC Data

As mentioned in section 4.1, non-IPOC measurements are stored differently from IPOC measurements. Since none of the other types of measurements are stored all at once at certain triggers, the timestamps of their measurements do not align. This is clearly visible in table 5.2, in which Continuous data of magnet A of beam B1 is shown, as well as the `BEAM_INTENSITY` of B1. Since the data is stored at different timestamps for each measurement, the feature dataset formed by joining these collections has a lot of misaligned data. After applying the forward fill approach described in subsection 4.1.1, the feature dataset contains the values displayed in table 5.3.

In order to create a feature dataset out of IPOC features and other features, the other features first need to be resampled so that they contain data at the timestamps for which there is IPOC data. This way, the features can be joined so that each data tuple in the feature dataset contains data for every feature.

[1] CERN has communicated that pulse generators can exhibit anomalous behaviour at any time, but there is not enough data (only the few State data collections) to add that study case here.

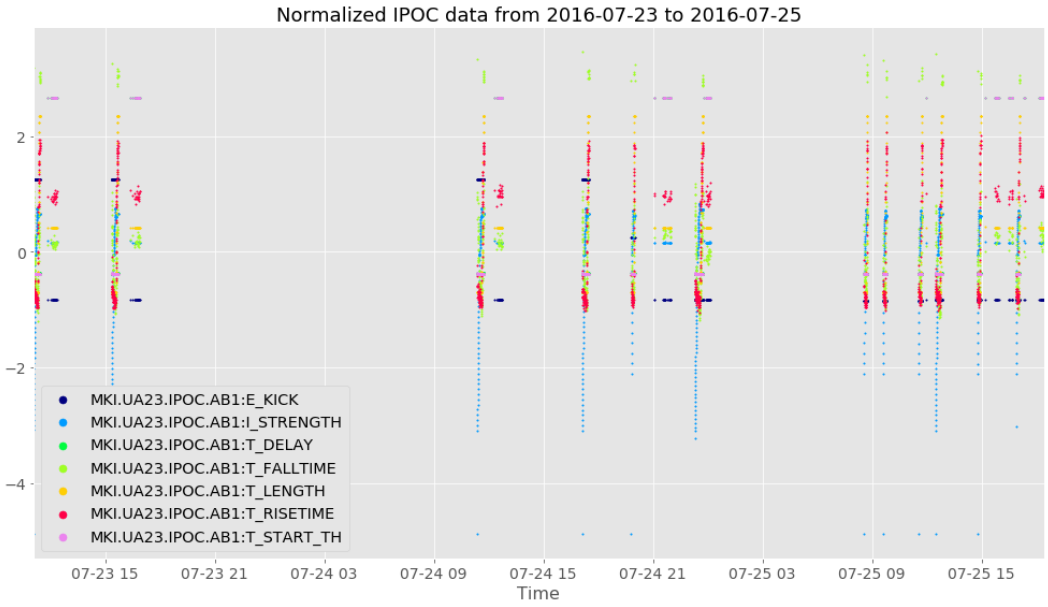


FIGURE 5.6: An even closer view of the last days of the data displayed in figure 5.5.

Timestamp	**...TUBE_DOWN**	**...TUBE_UP**	**PRESSURE**	**...INTENSITY**
2016-07-05 00:00:15.077	48.6	NaN	NaN	NaN
2016-07-05 00:00:17.013	NaN	NaN	NaN	2.256982×10^{14}
2016-07-05 00:04:21.133	NaN	95.1	NaN	NaN
2016-07-05 00:14:35.260	NaN	NaN	1.080000×10^{-10}	NaN
2016-07-05 00:15:15.269	48.8	NaN	NaN	NaN
2016-07-05 00:17:15.293	NaN	95.7	NaN	NaN

TABLE 5.2: A few Continuous data measurements of magnet A of beam B1 and LHC data measurements `BEAM_INTENSITY` of B1.

Timestamp	**...TUBE_DOWN**	**...TUBE_UP**	**PRESSURE**	**...INTENSITY**
2016-07-05 00:00:15.077	48.6	95.1	1.080000×10^{-10}	2.257001×10^{14}
2016-07-05 00:00:17.013	48.6	95.1	1.080000×10^{-10}	2.256982×10^{14}
2016-07-05 00:04:21.133	48.6	95.1	1.080000×10^{-10}	2.253294×10^{14}
2016-07-05 00:14:35.260	48.6	95.1	1.080000×10^{-10}	2.243816×10^{14}
2016-07-05 00:15:15.269	48.8	95.1	1.080000×10^{-10}	2.243158×10^{14}
2016-07-05 00:17:15.293	48.8	95.7	1.080000×10^{-10}	2.241397×10^{14}

TABLE 5.3: The feature dataset of table 5.2 after forward fill is applied. Note that `BEAM_INTENSITY` after the application of forward fill takes values other than the 2.256982×10^{14} in table 5.2. This is because there is much more `BEAM_INTENSITY` data between the timestamps in the table than the other three collections. They were omitted in order to keep the table short.

Before the resampling is done, all timestamps are rounded down from the millisecond to the second. This is done so that data can be resampled to seconds instead of milliseconds, which would create much more data. Rounding the timestamps doesn't cause any loss of data, because the data contains at most one measurement per second anyway. After the rounding, the non-IPOC features are resampled so that each feature contains a measurement for every second, and missing data is filled as in table 5.3. Then, to form the feature dataset, the IPOC data is joined with the resampled and filled non-IPOC data. This means that all of the IPOC data is used, along with all of the non-IPOC data at the timestamps of the IPOC data. The Continuous and LHC data collections are added to the feature dataset in this way.

Temporal Features

Beside the data collections themselves, transformations of collections are used as features as well. This is done to add more information to data tuples about the temporal relationship between measurements. The anomaly detectors examined in this thesis will view data tuples individually, without regard to neighboring tuples. However, there is a high correlation between measurements in the time dimension that is not visible in individual data tuples. The goal of features is to make detectors more effective at learning the temporal behaviour of magnets.

The potential benefit of temporal features can be illustrated as follows. Data exploration has shown that temperature measurements change relatively slowly compared to other measurements. If successive temperature measurements were to suddenly change by a large amount, this could be the result of anomalous behaviour. However, anomaly detectors that view data tuples individually would not catch this if the temperatures were still within their normal range. Temporal features would help show that the measurements have changed by a lot recently.

Sliding window features are created from the other features based on sliding window methods. These are the same ones that were used in the previous thesis [24]. The first feature, $SW1$, is created by taking the difference between each measurement and the average of a window of measurements before the measurement. This allows for sudden large changes in measurements to be detected. The second sliding window feature, $SW2$, is simply the sum of a window of measurements. This feature was created to handle divergent behaviour that occurs for longer periods of time rather than sudden large changes. If multiple successive measurements are abnormal, this will be caught by a significant difference in $SW2$.

The introduction of these new features brings a new parameter along with them.

Definition 5.1.1. The parameter **`sliding window size`** determines the size of the window to use for the creation of sliding window features.

It is important for this window size to be set to a value that will make the temporal features carry useful information. The window must be large enough so that measurements can be compared to the temporal behaviour of the segment they lie in, but not so large that information outside segments is included in the temporal features. Based on the discussion on segment lengths in subsection 4.4.4, a **`sliding`**

`window size` of 10 minutes seems like a good value. Since most segments run for 10 to 20 minutes, a window of 10 minutes will cause the most segments to carry temporal information about their own behaviour without including behaviour of other segments.

The two sliding window features, $SW1$ and $SW2$, are created for each of the Continuous and LHC features. They are calculated after the measurement timestamps are rounded and the collections are resampled and filled. Then, the temporal features at IPOC measurement timestamps are added to the feature dataset in the same way Continuous and LHC features are.

A feature dataset consisting of IPOC, Continuous, LHC, and temporal features will be created separately for each beam, as discussed in subsection 4.2.1.

5.1.3 Feature scaling

Some machine learning models, such as Gaussian Mixture Models, are sensitive to the varying ranges of values that different measurements have [24]. For instance, while temperatures can lie between 18 °C and 120 °C, `BEAM_INTENSITY` can go from 0 C to 2.5×10^{14} C. To make such models more effective at learning from the data, an optional feature scaling step is added to the pipeline.

Each feature vector in feature dataset can be scaled by centering it to its mean and scaling it to unit variance as follows.

$$\boldsymbol{x}' = \frac{\boldsymbol{x} - \mu}{\sigma} \tag{5.1}$$

where $\boldsymbol{x}$ is the feature vector, μ its mean, and σ its standard deviation. This manner of scaling is called standardization [10].

In order to be able to measure the effect that feature scaling has on different detectors, the parameter `scale data` was introduced.

Definition 5.1.2. The parameter `scale data` determines whether or not feature scaling will be applied to feature dataset before it is passed to an anomaly detector. It can be True (1) or False (0).

5.2 Anomaly Detection

The anomaly detection step is fairly straightforward: the feature dataset built in the previous step is passed to an anomaly detector, that detector is trained on the data, and is then used to predict an anomaly score for each data tuple in the dataset. When anomaly scores are generated for data tuples, they are appended to the tuples so that the dataset can be passed as a whole to the next step in the pipeline. Data extended with anomaly scores will be referred to as scored data.

For this thesis, the Isolation Forest and Gaussian Mixture Model algorithms were used as anomaly detectors. How they are trained and used to generate anomaly scores is explained in section 2.3. Section 6.3 describes how the source code developed

for this thesis allows for the implementation of new anomaly detectors to be used in the pipeline.

Definition 5.2.1. The parameter `anomaly detector` determines the model to be used as anomaly detector in the pipeline run. It can equal "isolation forest" or "gmm".

5.2.1 Hyperparameters

Hyperparameters are parameters that are given to anomaly detectors before the training process begins. While parameters such as `scale data` can transform the input dataset and thus indirectly influence the performance of detectors, hyperparameters influence the performance of detectors directly, by changing the way the detector model is trained. Each anomaly detector has its own set of hyperparameters. The most important ones for each of the detector models used are explained below. Hyperparameters can drastically affect the performance of detectors.

Isolation Forest

- **`n_estimators`:** the number of isolation trees.
- **`max_samples`:** the number of randomly chosen data tuples each isolation tree will be trained on.
- **`max_features`:** the number or ratio of features to draw from each data tuple when training isolation trees. The ratio is 1 by default.
- **`contamination`:** the prior ratio of anomalies in the dataset. This parameter is used by the algorithm to set a anomaly score threshold for which anomaly predictions will be made. This parameter is not used, as the evaluation approach described in section 5.4 works on anomaly scores.

Gaussian Mixture Model

- **`n_components`:** the number mixture components.
- **`covariance_type`:** the type of covariance parameters to use.
- **`init_params`:** the method to use to initialize weights, means, and precisions.
- **`n_init`:** the number of initializations to perform. The best results are kept.

Since most of the hyperparameters above are either integers or real numbers, an infinite amount of combinations of parameter values are possible. Subsection 5.4.3 introduces the approach that was used to find relatively well performing hyperparameters for the anomaly detectors.

5.2.2 Anomaly Score Transformation

In the implementation that was used, the Isolation Forest algorithm returns anomaly scores in the range $[-1, 1]$, where lower scores are more anomalous. Figure 5.7 shows an example of Isolation Forest anomaly scores for the data throughout 2015 and

2016 with hyperparameters `n_estimators` = 100 and `max_samples` = 512. The blue line in the figure consists of the scores sorted in increasing order, it illustrates the distribution of the scores. Only a small portion of the scores is less than 0, which is to be expected, because anomalous behaviour is rare.

An example of GMM anomaly scores with hyperparameters `n_components` = 2 and `covariance_type` = *tied* is displayed in figure 5.8. GMM anomaly scores are actually log probabilities which represent the likelihood of data tuples being normal. Therefore, it is also the case for GMMs that lower anomaly scores are used for more anomalous data. GMM anomaly scores are exceptionally different from Isolation Forest scores, whith a much smaller portion of the scores being more anomalous, and more anomalous scores being orders of magnitude different from normal scores. The most anomalous scores are so small that the structure of the scores is invisible in the figure.

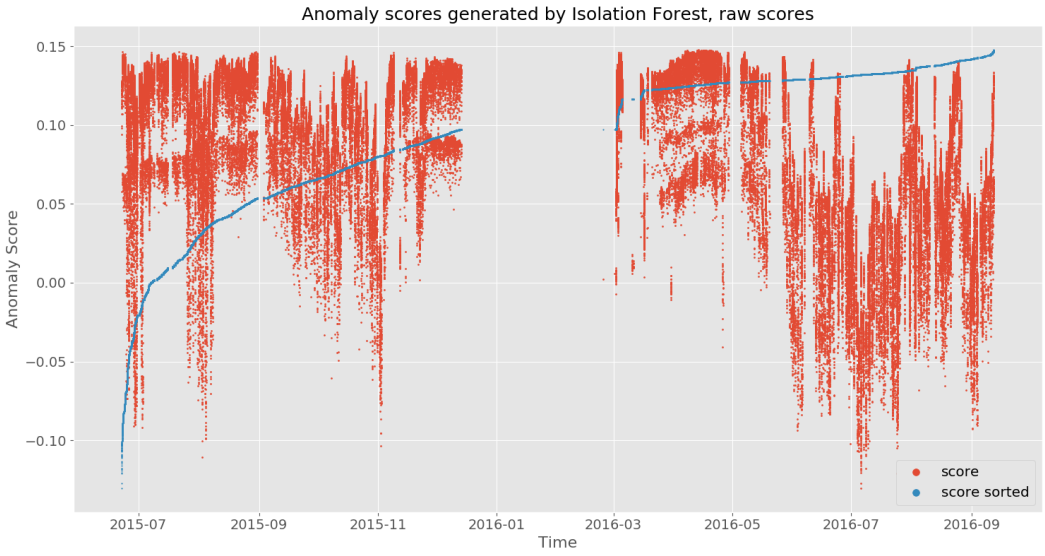


FIGURE 5.7: Raw anomaly scores generated by an Isolation Forest.

To keep the evaluation procedure clear and consistent for all anomaly detectors, anomaly scores are transformed to lie in $[0, 1]$, where 0 represents normal behaviour, and 1 represents anomalous behaviour. Transformed GMM anomaly scores can be seen in figure 5.9. The scores have been cut off at 0.05 so that the structure of the rest of the scores can be seen.

Table 5.4 contains an example of a scored feature dataset.

5.2.3 Dummy Anomaly Detectors

Dummy anomaly detectors were created in order to compare the performance of anomaly detectors to the performance of simple prediction strategies.

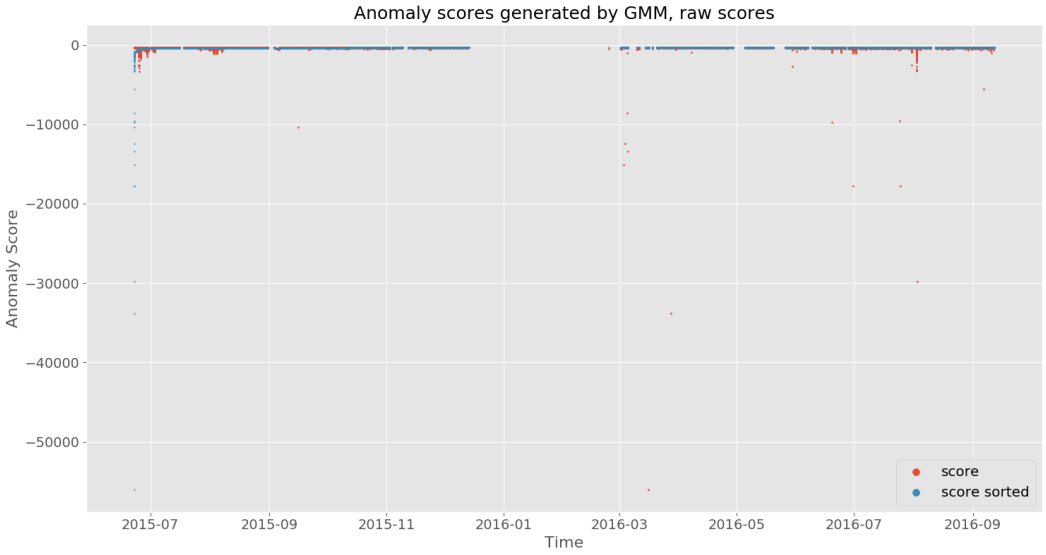


Figure 5.8: Raw anomaly scores generated by a GMM.

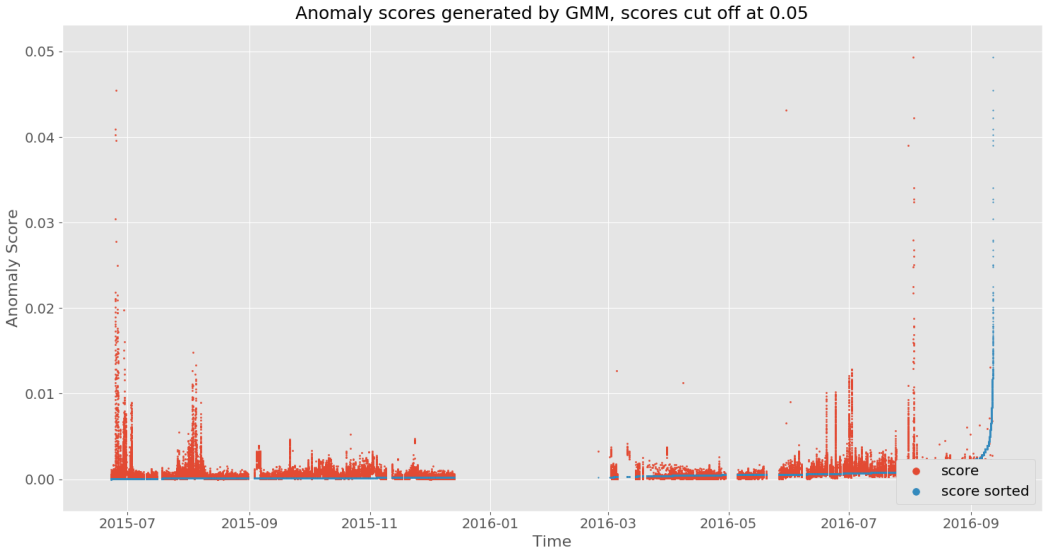


Figure 5.9: Transformed anomaly scores generated by a GMM, cut off at score 0.05 so that the structure of rest of the scores can be seen.

Timestamp	Feature 1	...	Feature N	Anomaly Score
00:00:00	...	...	...	0.58
00:01:00	...	...	...	0.68
00:02:00	...	...	...	0.12
00:20:00	...	...	...	0.79
00:21:00	...	...	...	0.03
00:22:00	...	...	...	0.46
00:53:00	...	...	...	0.85
00:54:00	...	...	...	0.99

TABLE 5.4: An example scored feature dataset.

Constant Detector

The constant Dummy detector returns anomaly scores equal to some constant value given to the detector.

Uniformly Random Detector

The uniformly random Dummy detector returns anomaly scores drawn randomly from a uniform distribution over $[0, 1)$. An example of this detector's anomaly scores is shown in figure **??**.

Stratified Random Detector

The stratified random Dummy detector draws its anomaly scores randomly from two uniform distributions, one over $[0, 0.7)$, the other over $[0.7, 1)$. It takes a `contamination` parameter, which determines the ratio of scores to be drawn from the second distribution. This is done to simulate the effect of assigning higher scores to occasional anomalous behaviour. Figure 5.10 displays a set of anomaly scores generated with `contamination` set to 10%.

5.3 Postprocessing

Before the scored feature dataset is passed to the evaluation step, postprocessing is applied to make evaluation a more approachable task. The scored data is transformed into a list of annotated IPOC segments.

5.3.1 Evaluation Approach

To understand why a postprocessing step is necessary, the evaluation approach must be understood first. It is explained here after two other possible approaches are discussed. Recall that subsection 4.1.6 explained that a lot of uncertainty comes with anomaly labels; the actual anomaly that a label was created for could lie anywhere in

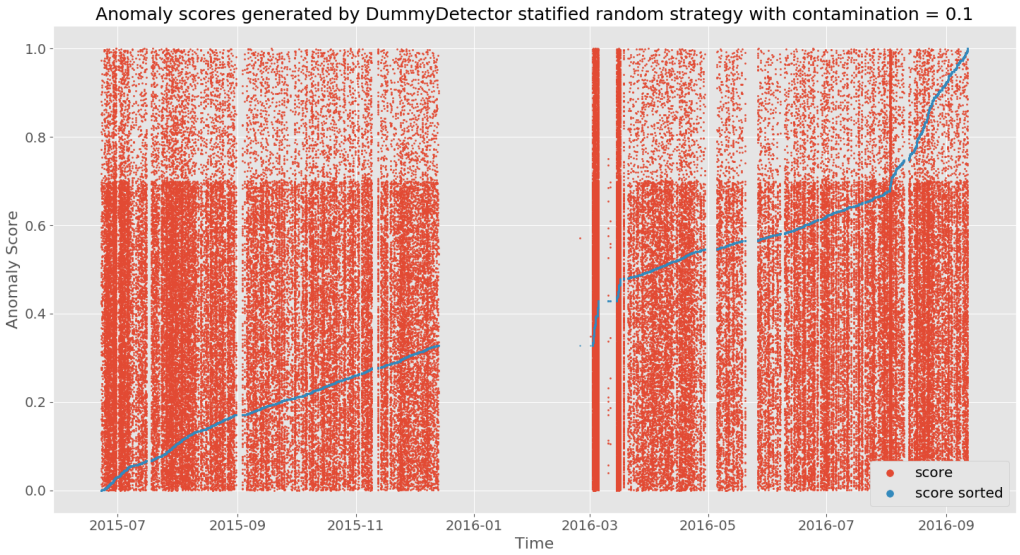


Figure 5.10: Anomaly scores generated by a Dummy detector with the stratified random strategy.

a region of 12 hours before the label's timestamp. It is highly likely that anomalous behaviour manifests itself in the data for only a small section of such a region.

A simple approach towards evaluation would be to mark all data tuples in the regions of 12 hours before anomaly labels as anomalous and to then check that those anomalous data tuples were assigned a higher anomaly score than normal data tuples. However, this approach is not a good option. It would cause all detectors to perform poorly, because most of those data tuples marked as anomalous would not actually be anomalous, but normal. This approach would cause high anomaly score expectations for many data tuples for which low anomaly scores should be expected.

Another approach would be the one developed by Wéry in the previous thesis [24], where a predetermined amount of worst anomaly scores is selected and grouped into "anomalous segments" based on how close the scores lie near each other in time. Evaluation is then done by checking the ratio of anomalous segments that lie in the region of an anomaly label. This approach has a few noticeable drawbacks. Firstly, it causes the evaluation step to be dependent on a "top K" parameter, (the amount of worst scores to take). Secondly, there is no consistent basis of ground truth, which makes comparisons between different detectors, and even different runs of the same detectors, impractical. The need for a consistent basis of ground truth is discussed in subsection 5.3.3.

The evaluation approach developed for this thesis instead relies on IPOC segments. First, scored feature datasets are transformed into lists of segments. Anomaly scores of segments are then calculated using the scores of the data tuples in the segments. Labels are used to determine which segments are normal and which are anomalous, represented by 0's and 1's, respectively. Finally, the anomaly scores and labeling of

segments are used to calculate the performance of detectors as explained in section 5.4. This approach represents the performance of detectors well, is not dependent on a "top K" parameter, and can be used to compare different detectors.

Note that while the new evaluation procedure does not rely on a "top K" parameter, it does on `segmentation distance`. However, the segmentation based on `segmentation distance` tries to catch a real world phenomenon, the existence of machine usage sessions. IPOC segments are not just collections of data tuples, they have a semantic meaning. Also, all data tuples will always be grouped into segments, as opposed to only grouping a certain amount of tuples that have the worst anomaly scores. Furthermore, in the "anomalous segments" approach, it is possible for single data tuples with extraordinarily high anomaly scores to be flagged as anomalies. However, actual anomalous behaviour will most likely occur in multiple data tuples that occur closely after each other. By using IPOC segments, this fact can also be taken into consideration, as is explained in the introduction of segment anomaly scores in the following subsection.

The abstraction level of IPOC segments is what allowed for the new evaluation approach to be developed. It is the reason why they were introduced and why the problem statement was adapted (in subsection 4.4.3) to make the goal the detection of anomalous segments rather than anomalous behaviour at any given moment.

5.3.2 Segmentation

The scored feature dataset created by the anomaly detection step in the pipeline is transformed into a list of segments as described in subsection 4.4.1. As described above, it is likely that anomalous behaviour only occurs for short periods at a time in the data. Anomalous segments were defined as segments that contain anomalous behaviour. A good anomaly detector will assign those data tuples that belong to anomalous behaviour high anomaly scores. Therefore, anomalous segments should contain at least a few data tuples that have high anomaly scores, and normal segments should contain mostly low scores.

Segment Anomaly Score

Many definitions for the anomaly score of a segment are possible. For this thesis, three methods were defined and tested. All three of them try to catch the fact that anomalous segments should contain at least a few high anomaly scores.

The segment anomaly score methods are defined as follows:

- **Max:** The anomaly score of a segment is equal to the maximum anomaly score of the data tuples the segment is composed of.
- **Top K:** The anomaly score is equal to the average of the K worst anomaly scores in the segment. For this thesis, K was set to 10.
- **Top Percentage:** The anomaly score is equal to the average of a percentage of the worst anomaly scores in the segment. For this thesis, the percentage was set to 25%.

Definition 5.3.1. The parameter **`anomaly score method`** determines the definition to use for the calculation an IPOC segment's anomaly score. It can take values "max", "top_k", and "top_percentage".

5.3.3 Ground Truth Annotation

In order to be able to compare performance of different anomaly detection pipeline runs, a consistent basis of ground truth must be present in the data. What that means in the context of this thesis is that in the evaluation step, the segment anomaly scores that result from anomaly detection must be compared to the same expected values every time.

The ground truth consists of the labels described in subsection 4.1.6. This sub-step is called ground truth annotation because segments generated by the segmentation of the scored feature dataset are annotated with labels. This is done by checking for each label whether or not it lies in the 12 hour region of a label, and annotating the segment with the label if it does. An example of this can be seen in table 5.5. It shows the segmented and annotated version of Table 5.4 with parameters `segmentation distance` = 15 min and `anomaly score method` = max.

Segment	Anomaly Score	Label
1	0.68	none
2	0.79	intervention
3	0.99	anomaly

Table 5.5: An example scored and annotated segment list.

Subsection 4.1.6 introduced the notion that machine behaviour caused by interventions or research could also be viewed as anomalous from the point of view of the data. Because of this, ground truth annotation will be done with only anomaly labels and with anomaly, intervention, and research labels. In the output of the pipeline, `types_of_labels` will denote which labels were used for annotation. It will equal "anomaly" if only anomaly labels were used, and "any" if all 3 mentioned types were used.

Definition 5.3.2. The parameter **`labels`** determines the type of labels to be used in the ground truth annotation of segments. It can be either "anomaly" or "all".

5.3.4 Labeled Segment Length

To follow up on the discussion of segment length in subsection 4.4.4, figures 5.11 and 5.12 display the distributions of the lengths of labeled and anomalous segments, respectively. Labeled segments are segments that lie near any label that is not an info label. Anomalous segments are labeled segments for which the label is an anomaly label.

The length distributions for both labeled and anomalous segments look very similar to those of all segments, which indicates that the segments that should be

detected are approximately of the same length as normal segments. Segment length appears to be irrelevant to the appearance of anomalous behaviour.

While the distributions are very similar to those in subsection 4.4.4, these ones are much less smooth, especially those of anomalous segments. This is not caused by a different pattern in the distributions, but by density estimation being done on fewer segments. Note that there are more than 10 times fewer labeled segments than the total amount of segments, and that there are even fewer anomalous segments.

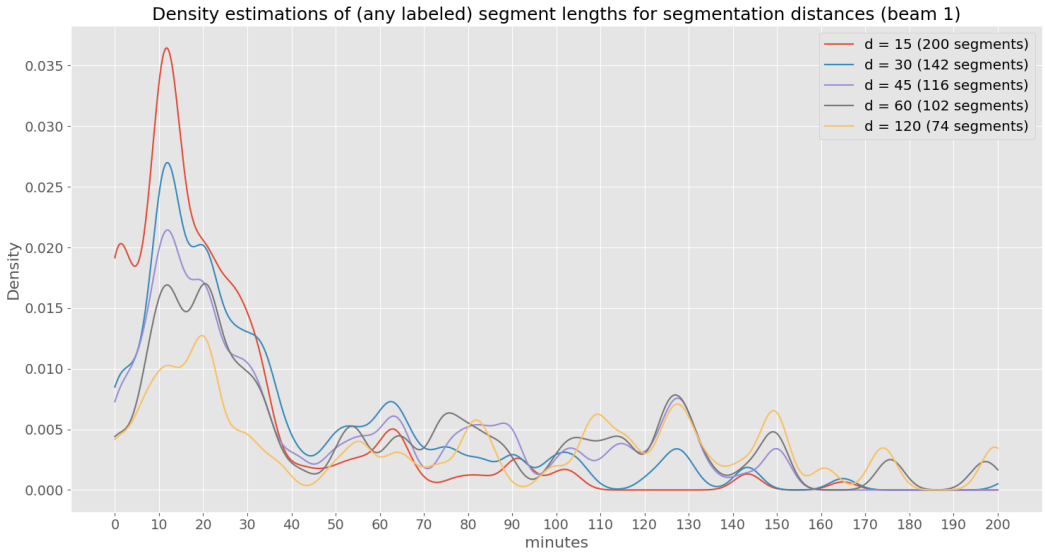


FIGURE 5.11: Distributions of lengths of labeled segments for different segmentation distances.

5.3.5 Labeled Segment Pruning

In ground truth annotation, labels are linked to segments that lie in the 12 hour regions before the timestamps of the labels. The segments that have labels are then those that should be detected as anomalous by anomaly detectors. There is a slight problem with the way segments are annotated as described above. Similarly to how marking all data tuples in the 12 hour label regions wrongly marks way too many tuples as anomalous, annotating every segment that lies in those regions will often mark multiple segments as well. Most likely, only one segment in a label region contains the anomalous behaviour that caused the label to be created.

Figure 5.6 helps to illustrate this. Imagine there was an anomaly label with timestamp "07-25 15:00:00". That label would then be linked to all of the segments between "07-25 03:00:00" and "07-25 15:00:00", which would cause the evaluation step to expect high anomaly scores for all of those segments, while in reality only one of them was anomalous.

Having multiple segments annotated with the same label will cause evaluation metrics to represent the performance of anomaly detectors incorrectly. While the

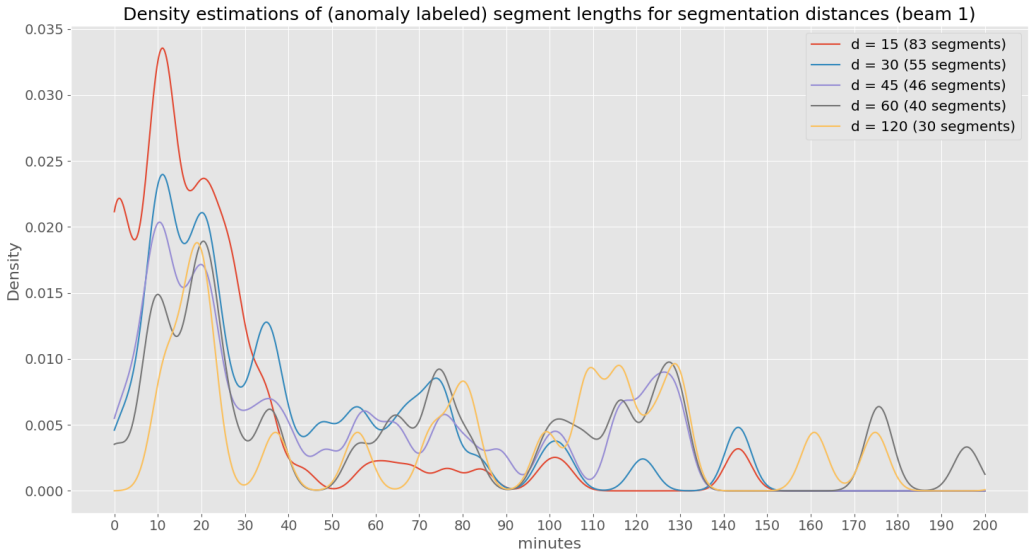


Figure 5.12: Distributions of lengths of anomalous segments for different segmentation distances.

metrics will not be affected as much as they would be by using the first evaluation approach that was mentioned, they will still be affected to a certain extent. Labeled segment pruning is introduced to combat this effect. For each label, the set of segments that the label is linked to is replaced by the segment with the highest anomaly score. This way, each label carries a weight of only one segment in the evaluation step.

5.4 Evaluation

In order to determine the performance of an anomaly detector, a formal evaluation procedure must be established. While the anomaly detectors in question are unsupervised machine learning models, evaluation happens in a supervised way by using the electronic logbook labels. A part of the evaluation approach is already explained in subsection 5.3.1. This section explains the evaluation metric that was chosen and how the pipeline supports experiments with all parameters to be done.

5.4.1 Interpretation of Anomaly Scores

At this point, the raw data has been processed into a list of labeled IPOC segments with anomaly scores. All that needs to be done to evaluate these anomaly scores is to turn the scores into predictions that state whether a segment is anomalous (1) or normal (0). This is simply done by setting a threshold on the scores. If a segment's anomaly score exceeds this threshold, it is predicted by the anomaly detector the be anomalous, otherwise it is predicted to be normal. Different thresholds will result in

different performances. For example, if the threshold is set to 0, all segments will be predicted as anomalous, which is obviously wrong. A threshold of 0.999 might cause only truly anomalous segments to be predicted as anomalous, but it might be too restrictive and cause many anomalous segments to be falsely predicted as normal. In performance calculations, the performance metric will be calculated for all possible anomaly score thresholds and a plot will be made that shows the metric for every threshold. This way, a threshold can be chosen without having to test different ones manually.

In one-class classification problems, it is generally the case that the phenomenon or object being classified is referred to as the "positive class". Since the anomaly detection problem at hand is effectively a one-class classification problem, anomalous behaviour will be referred to as the positive class, and normal behaviour as the negative class. This then also makes the meaning of TP, FP, TN, and FN (as defined in section 2.4), clear in the context of anomaly detection. These terms will be used to calculate classification performance.

5.4.2 Performance Metric

Since the pipeline will be ran many times with different combinations of parameters, a single evaluation metric that can be used to compare the quality of drastically different parameter settings is desirable. Three widely used performance metrics are discussed in this subsection, area under the ROC curve (AUROC), area under the PR curve (AUPR), and F score. All three of these metrics are based on ratios between true positives, false positives, true negatives, and false negatives. They have been defined in subsection 2.4.

Metrics such as TPR, FPR, Precision, and Recall are undesirable because they only represent performance in one sense. For example, TPR and Recall both calculate the ratio of actual positives which are correctly identified as such, but they state nothing about the ratio of actual positives in the set of positive predictions (which is calculated by Precision). Instead, metrics that are combinations of other metrics are discussed so that one number can be used to represent performance.

Davis and Goadrich state in [9] that *"... with highly skewed datasets, Precision-Recall (PR) curves give a more informative picture of an algorithm's performance."* and that *" ... algorithms that optimize the area under the ROC curve are not guaranteed to optimize the area under the PR curve."*. They further explain that this is due to the fact that the amount of negatives greatly exceed the amount of positives in the ground truth. Because of this, false positive rate (FPR), which is used in ROC curves, changes only slightly even when there is a large change in the amount of false positives. Alternatively, precision, which is used in PR curves, changes much more strongly in this context. PR-curves omit the issue created by the large amount of negatives by not using the number of true negatives in their calculations.

Following the reasoning above, PR curves will be calculated to portray anomaly detection performance. Different performance results will be compared by the area under the curve; the higher the area, the better the performance. Another benefit of PR curves is that all (precision, recall) tuples for different thresholds are visible on

the curve. The result of this is that a threshold can be chosen by CERN depending on the costs of false positives and the costs of false negatives.

The F score metric calculates a balance between the precision and recall for a certain threshold. It is not used, as it essentially conveys the same performance as a PR curve essentially conveys, without the added bonus of being able to see all (precision, recall) tuples.

5.4.3 Grid Search

The final pipeline that was developed takes various parameters to run. A brief recap of them and where they are introduced: `segmentation distance` (definition 4.4.2), `scale data` (definition 5.1.2), `anomaly detector` (section 5.2.1), `hyperparameters` (subsection 5.2.1), `anomaly score method` (definition 5.3.1), `labels` (definition 5.3.2).

Each of these parameters has an influence on the pipeline's resulting AUPR and thus has to be tuned so that anomaly detection can be performed as well as possible. Note that due to anomaly detector hyperparameters, a large portion of the parameter space cannot be tested, as many of the parameter combinations would require enormous computational and memory resources. This is the case for large values for `n_estimators` with Isolation Forest and for `n_components` with GMM. Fortunately, the amount of combinations of the other parameters is much more limited in comparison, as most of them can only take a few different values, and the `segmentation distance` range should be limited to $[30, 60]$ as discussed in subsection 4.4.4.

Parameter optimization is done using a grid search approach, in which a grid of parameters is set up and the predetermined combinations of parameters are used exhaustively. For each combination of parameters, the resulting AUPR is saved so that the best performing parameters can be returned.

All of the pipeline parameters can be grouped into two types: training parameters and evaluation parameters. Training parameters are those parameters that influence the trained anomaly detectors models. They are `scale data`, `anomaly detector`, and `hyperparameters`. Evaluation parameters are `segmentation distance`, `anomaly score method`, and `labels`. These parameters don't affect the anomaly detectors, but the way their resulting anomaly scores are evaluated. The consequence of splitting these parameters into two groups is that grid search can be executed more efficiently: anomaly detectors are trained using combinations from the training parameter grid, and each time a detector is trained, it is evaluated using all combinations of the evaluation parameter grid.

Appendix C shows an excerpt of the output of a grid search run. In the excerpt, a GMM is trained for a combination of training parameters and evaluated for all combinations of evaluation parameters.

In the Python software package developed for this thesis, grid search with all parameters can be executed using one simple command as shown in example 5.4.1. The source code itself is discussed briefly in section 6.3.

Example 5.4.1. A `grid_search` function call that executes the anomaly detection pipeline for GMM with different training and evaluation parameters. The parameters that are assigned lists in this function call will take each of the given values at different points in the parameter grids produced by the lists.

```
results = pipeline.grid_search(
    features=df,
    labels=[labels_anomaly, labels_all],
    anomaly_detector="gmm",
    detector_parameters={
        "n_components": [2, 4, 6, 8],
        "covariance_type": ["full", "tied"],
        "init_params": ["kmeans"],
        "n_init": [1],
        "verbose": [1]
    },
    scale_data=[True, False],
    segmentation_distance=[10, 20, 30],
    anomaly_score_method=["max", "top_k", "top_percentage"],
    verbose=True,
    show_figures=True,
    fig_filename_func=get_filename,
    output_filename="grid_search_gmm"
)
```

Chapter

6 Results

This chapter discusses anomaly detection performance that was achieved using the pipeline detailed in chapter 5. Then, a concise overview of what has been done differently compared to work done by Wéry [24] is given. Lastly, the source code developed for this thesis is described at a high level.

6.1 Anomaly Detection Performance

Dummy detector performance is reviewed first, so that an image can be given of how well other detectors compare to simple strategies that do not try to learn what constitutes normal behaviour.

6.1.1 Dummy Detectors

Since most IPOC segments contain normal behaviour, random Dummy strategies (introduced in subsection 5.2.3) have a relatively consistent performance. The precision metric is always very low, causing a low area under the PR curve. This is the result of random strategies returning way too many positives (which is certainly the case for the uniform strategy or stratified strategies with high contamination) and the positives being distributed evenly in time. The combination of these two factors results in a high number of false positives and a low number of true positives.

As mentioned in subsection 5.4.3, the postprocessing and evaluation steps take parameters named evaluation parameters. Different values for these parameters cause the performance to vary. Figures 6.1 and 6.2 show PR curves for random Dummy detectors for two different sets of evaluation parameters. The legends of the figures show the AUPR for each strategy. The areas in figure 6.2 are significantly better for each strategy.

Note that no constant Dummy strategy is present in the PR curves. This is because no PR curve can be drawn for a constant strategy. If all anomaly scores are the same, e.g. 0 or 1, then there are no different options for anomaly score thresholds, all predictions will be either positives or negatives. Therefore, for each combination of evaluation parameters, only two (precision, recall) tuples can be calculated for constant strategies.

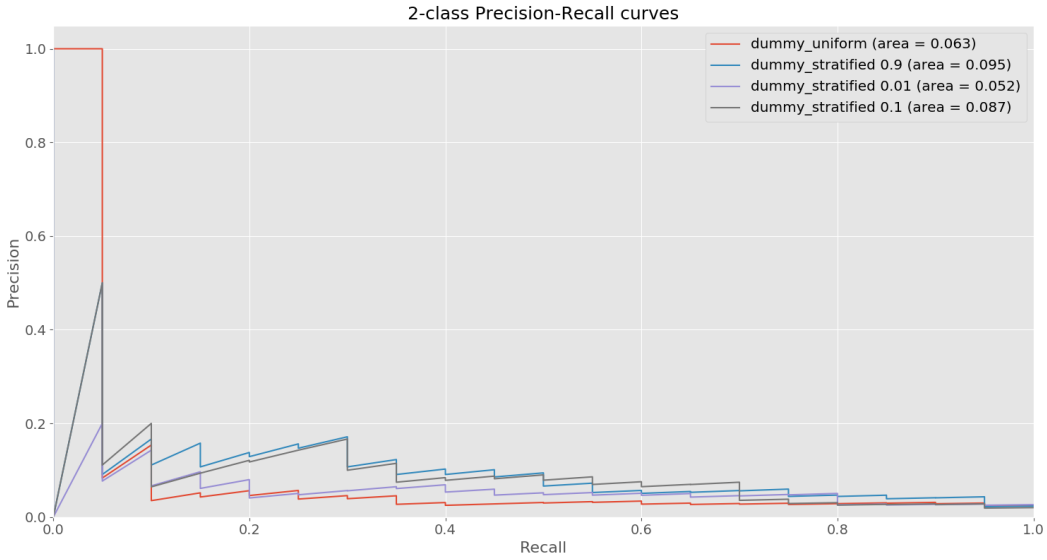


Figure 6.1: PR curves of Dummy detectors with evaluation parameters `segmentation distance` = 20 min, `anomaly score method` = max, `labels` = *anomaly*.

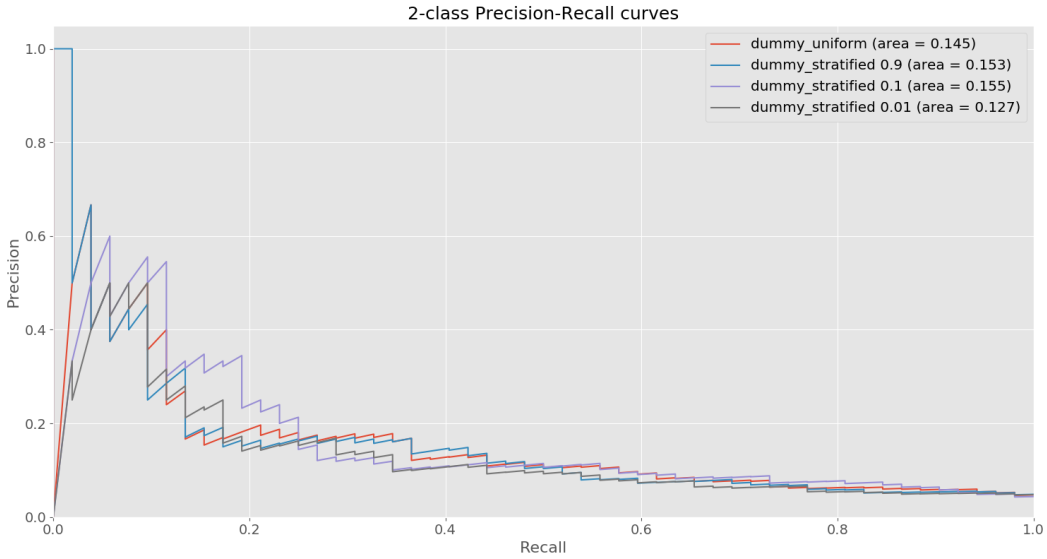


Figure 6.2: PR curves of Dummy detectors with evaluation parameters `segmentation distance` = 30 min, `anomaly score method` = top_k, `labels` = *all*.

6.1.2 Grid Search Results

To find a good set of pipeline parameters and anomaly detector hyperparameters, grid search was executed iteratively. A relatively small grid of parameters was tested each time, and new grids of parameters for later grid search runs were chosen based on the performance of previously tested parameter values. This was done to save time, as executing grid search on a grid that contains many options for each of the parameters would take a tremendous amount of time and run many unnecessary pipeline executions with parameter combinations for which can be predicted that the performance would be poor. This subsection displays the best results that were found using this approach, along with a brief discussion on which values performed well for each of the parameters in the experiments that were done.

GMM

Figure 6.3 shows the best PR curve achieved with the GMM anomaly detector. Figure 6.4 displays the predictions that would be made if the 99-th percentile segment anomaly score would be chosen as the threshold. Labels and predictions are plotted over different data collections so that the predictions can be compared quickly to what was going on at the time in a few data collections. The 99-th percentile threshold in this case is 0.044, which agrees with the (precision, recall) tuple of (0.54, 0.37). Note that a better threshold could have been chosen here, the one that resulted in the tuple of about (0.78, 0.37) in figure 6.3. Interestingly, the predictions are nearly spot on for the data in 2016, but completely miss everything in 2015.

Isolation Forest

Figures 6.5 and 6.6 show the best PR curve achieved with the Isolation Forest anomaly detector and the predictions for the 99-th percentile anomaly score threshold of the PR-curve. Overall, the performance of the Isolation Forest detector was abysmally low, even worse than the Dummy detectors. However, this can be attributed to a pattern clearly visible in figure 6.6. The Isolation Forest seems to learn that high temperatures lead to anomalous behaviour, which is clearly wrong. Because of this pattern, an experiment with an adapted feature dataset was ran: the Isolation Forest was trained on a feature dataset consisting of only IPOC features. While the resulting AUROC was still much lower than that of the GMM detector, it did grow significantly, and this time performed better than Dummy detectors. The PR curve and resulting predictions made for this experiment can be seen in figures 6.7 and 6.8, respectively.

Parameters

What follows are brief notes on the impact of parameters that was observed after grid search runs.

- **`sliding window size`:** Definition 5.1.1. Tested 10 minutes and 30 minutes. Results for a sliding window of 10 minutes were always better. It seems that the

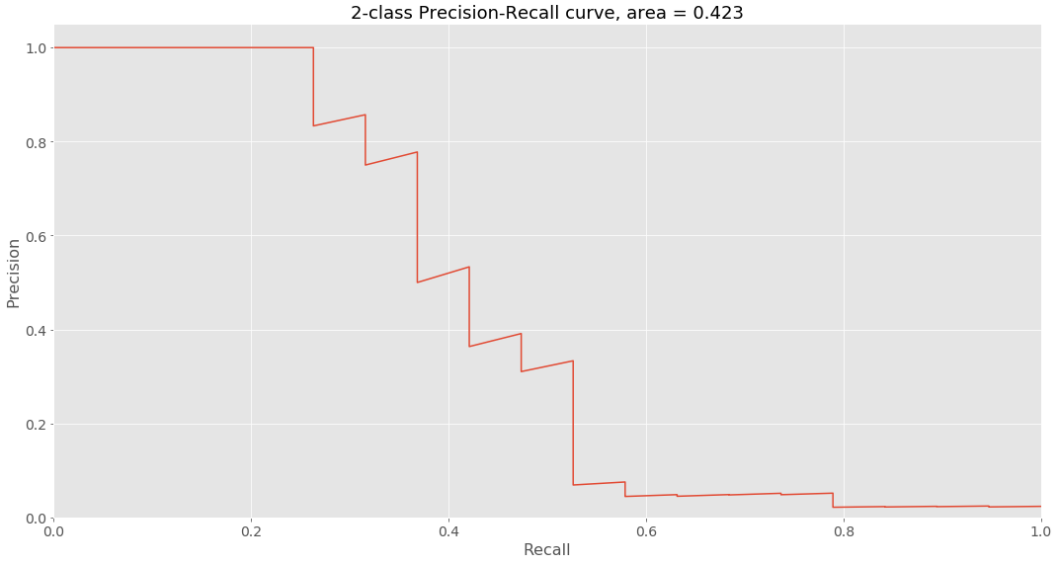


Figure 6.3: The best PR curve achieved with the GMM anomaly detector. The parameters were: `n_components` = 6, `covariance_type` = $full$, `scale data` = $False$, `segmentation distance` = 60 min, `anomaly score method` = top_k, `labels` = $anomaly$

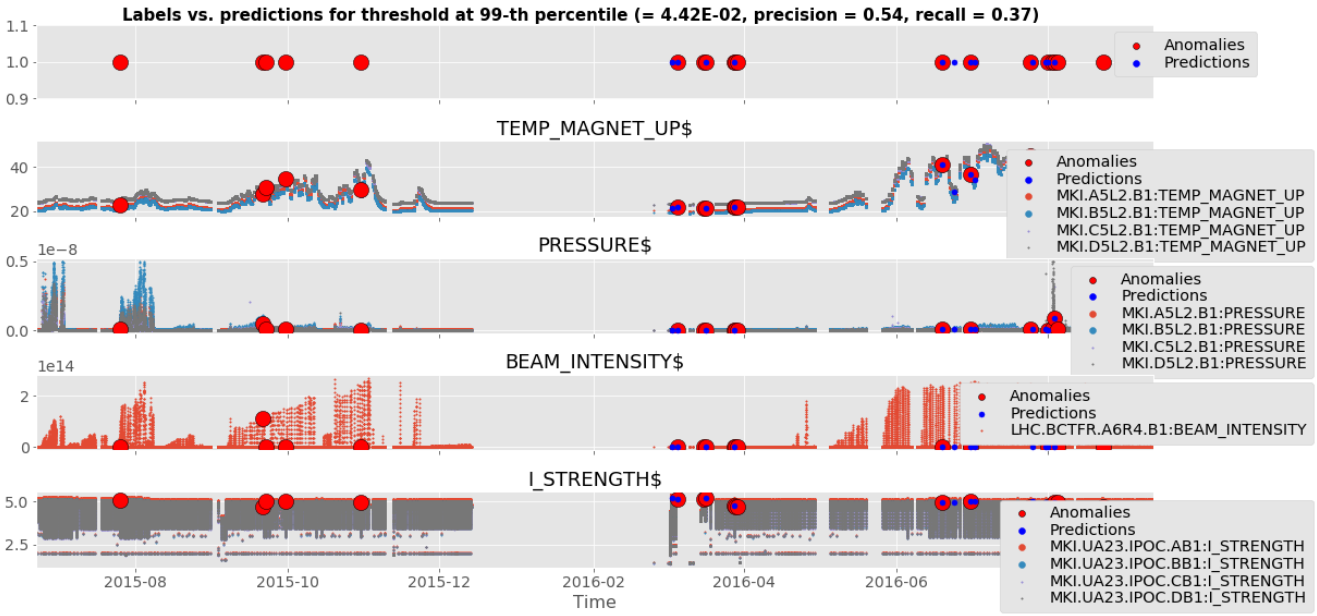


Figure 6.4: Predictions made for the 99-th percentile threshold of the PR curve in figure 6.3.

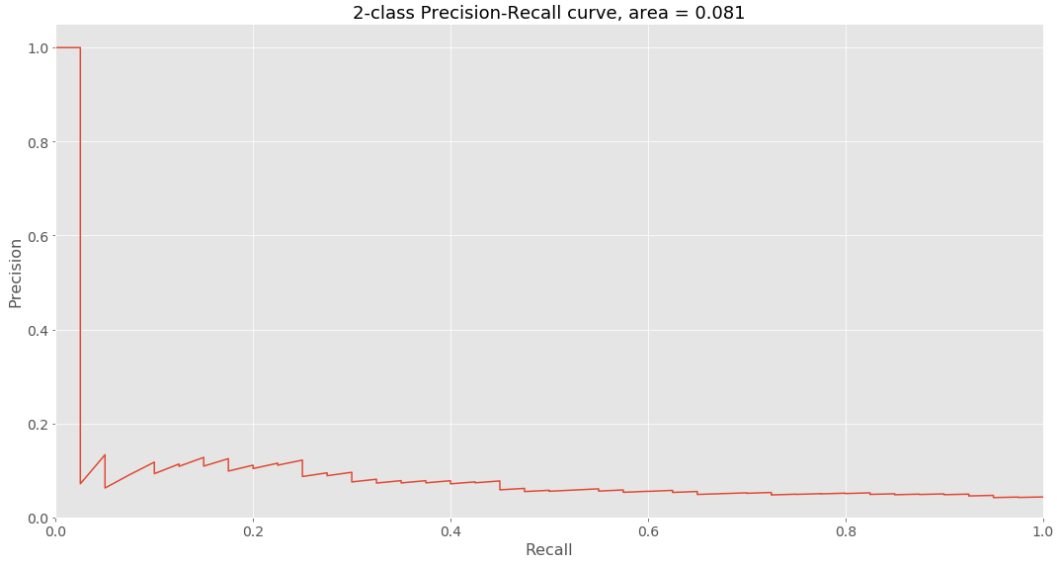


FIGURE 6.5: The best PR curve achieved with the Isolation Forest anomaly detector. The parameters were: `n_estimators` = 250, `max_samples` = 5120, `scale data` = *False*, `segmentation distance` = 60 min, `anomaly score method` = max, `labels` = *anomaly*

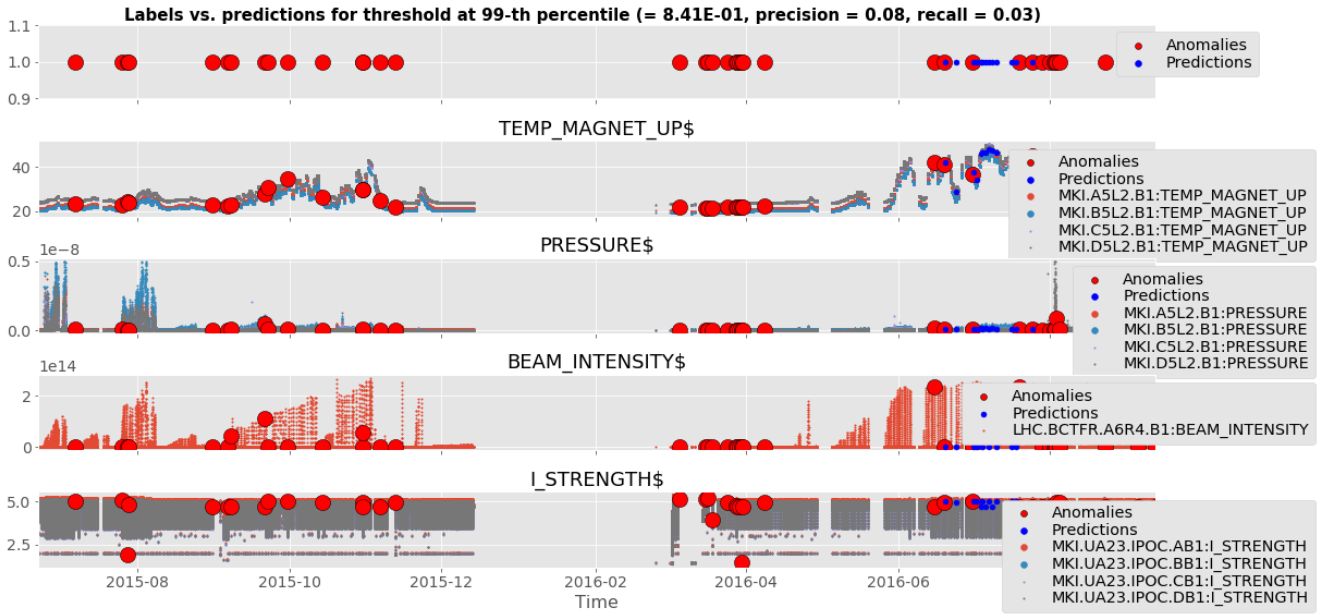


FIGURE 6.6: Predictions made for the 99-th percentile threshold of the PR curve in figure 6.5.

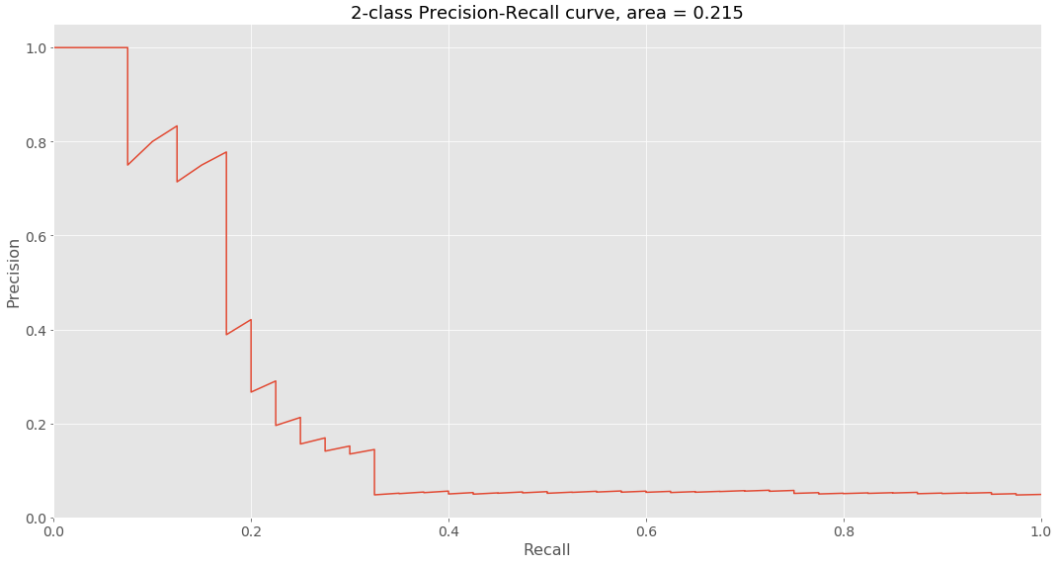


Figure 6.7: PR of the Isolation Forest anomaly detector with the adapted feature dataset. The parameters were the same as the ones in figure 6.5

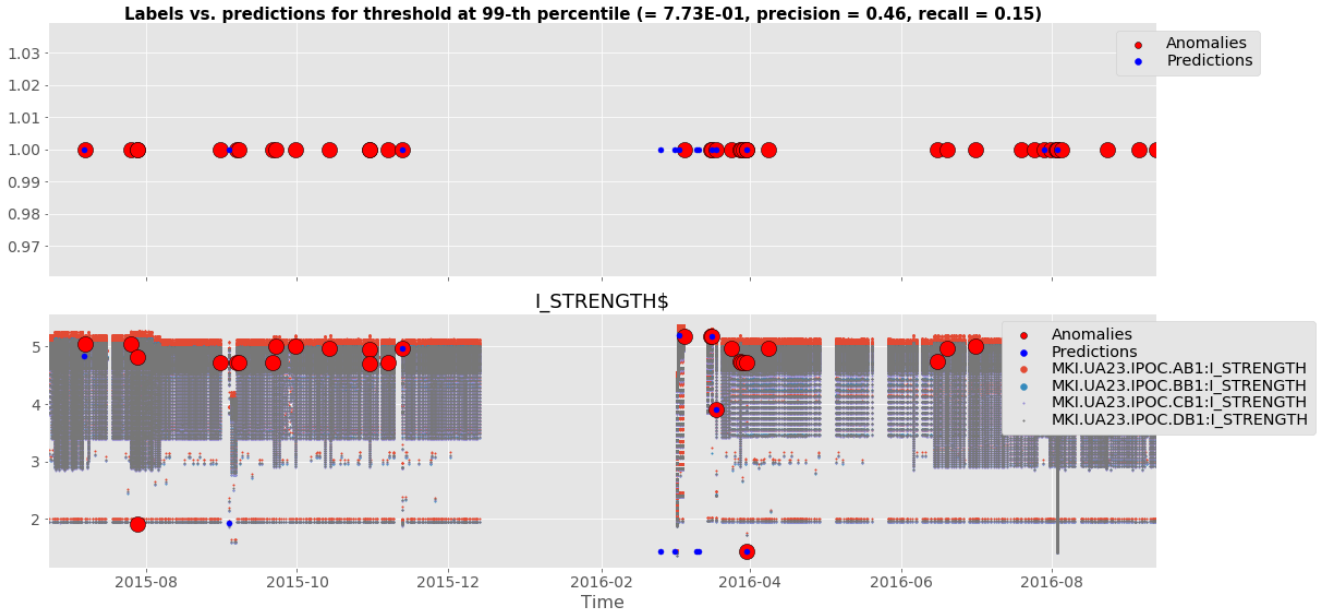


Figure 6.8: Predictions made for the 99-th percentile threshold of the PR curve in figure 6.7.

sliding window of 30 minutes does indeed take too much information outside of segments into account.

- **`segmentation distance`:** Tested values in [15, 60]. Higher distances result in better performance. Likely because those distances make IPOC segments lie closer to the machine usage sessions that took place in reality.
- **`scale data`:** Has no impact on either of the anomaly detectors, even though GMMs were mentioned to be sensitive to data distributions in subsection 5.1.3.
- **`anomaly score method`:** "top_k" performs the best, followed closely by "max". This confirms the reasoning behind the methods explained in subsection 5.3.2. "top_percentage" performs significantly worse, because it takes too many normal data tuples into the anomaly score calculation.
- **`labels`:** "anomaly" generally performs better. This refutes the claim made in subsection 5.3.3 that the other events could be viewed as anomalies. It does seem like anomalous behaviour is quite different from the others and that the models are better at distinguishing it.

GMM specific parameters:

- **`n_components`:** Tested [1, 8]. Larger numbers generally perform better, while in the previous work done using GMMs, 2 was proposed as a good option [24].
- **`covariance_type`:** Tested "full", "tied", "diag", and "spherical". "full" performed the best, while "sperical" performed the worst by far.
- **`init_params`:** Tested "kmeans" and "random". As is to be expected, "kmeans" performs better.
- **`n_init`:** Tested [1, 5]. Value didn't matter much when using "kmeans". Kept at 1 in most runs to save time.

Isolation forest specific parameters:

- **`n_estimators`:** Tested values in [100, 1250]. Limited to 1250 as more than that would not fit in the memory of my machine (16GB). Value didn't seem to matter, performance was always poor. Kept at low numbers in later runs for faster grid search runs.
- **`max_samples`:** Tested values in [256, 5120]. Higher was generally better. It seems that the isolation trees are less affected by the large temperature values when they learn from more smaples. More experiments with higher values should be done.
- **`max_features`:** not tested.

Note that it is possible for anomaly detectors to mark certain segments as anomalous because an anomaly did actually occur, but it was not detected by CERN, and thus no label was created. A way to check this would be to implement more anomaly detectors and check if multiple of them mark certain segments which are not annotated with a label as anomalous.

6.2 Previous Work

As explained in section 3.4, Wéry [24] created an anomaly detection application based on GMMs. A part of his evaluation approach has already been discussed in subsection 5.3.1. The output of trained GMMs was put through a novel segmentation algorithm so that predictions could be made about where anomalies occurred. The goal of the algorithm was to avoid the need for a top K anomalies parameter and to automatically group anomalous data tuples that lie nearby each other into segments so that one anomaly prediction is made instead of many. While it does make sense to view nearby data tuples with high anomaly scores as one anomaly, the algorithm allowed for anomalies to be predicted at any moment, also at times where the magnets were not in use. Moreover, while the algorithm didn't explicitly use a top K anomalies parameter to make predictions, it used a parameter that was essentially equivalent.

Wéry's anomaly segmentation algorithm was what inspired the idea of creating segments based on the timestamps of the input feature dataset rather than the output of a detector. It is more intuitive to split the data into segments of machine usage and to then predict whether segments are normal or anomalous using the anomaly scores of the data tuples that form the segments. This approach then allowed for the evaluation procedure explained in section 5.4 to be developed. The "top K anomalies" view was replaced by the "anomaly score threshold" perspective, which is very similar, but more directly applicable to the anomaly scores of segments. The search for the best anomaly score threshold was then handled automatically by PR curves, which display the performance of a detector using every possible threshold.

Although the evaluation procedure has been changed drastically enough that results cannot be compared, the main differences between the developed methods can still be listed:

- Replaced complex segmentation algorithm for anomaly detector output by a simpler and more intuitive segmentation algorithm based on IPOC measurement timestamps. Instead of calculating performance with "anomalous segments", it is now calculated using IPOC segments (subsection 5.3.1).
- Defined evaluation metrics in terms of TP, FP, TN, and FN, instead of ambiguous terms like "number of segments with labels" (subsection 5.4.2).
- Consistent basis of ground truth (subsection 5.3.3) allows for a more correct comparison of results.
- Instead of calculating Precision and Recall for the top K worst anomaly scores and thus implicitly setting a threshold on the anomaly scores, the metrics are calculated explicitly for every threshold and a PR curve is plotted (subsection 5.4.1).
- Made the training parameters configurable, introduced the evaluation parameters, and introduced a parameter optimization approach (subsection 5.4.3).
- DummyDetectors were added in order to compare machine learning models to simple random strategies (subsection 5.2.3).
- The two LHC data collections have been added as features (subsection 5.1.2).

6.3 Source code

The anomaly detection pipeline has been developed in a modular way so that the components of the pipeline can be extended easily by new implementations. The source code written for this thesis supports this as well. For example, the `anomaly_detection` module contains an `AnomalyDetector` interface that defines all of the functions an anomaly detector should implement so that it can be used in the pipeline. The interface can be seen in appendix D. When these functions are implemented for a new detector and the detector is added to the `AnomalyDetectorFactory`, it can simply be passed as a parameter to the pipeline code (as seen in example 5.4.1) in order to execute the whole pipeline and collect results.

The pipeline code was written so that it can be executed with any combination of training and evalaution parameters mentioned in subsection 5.4.3.

The most important modules of the source code are as follows:

- **`preprocessing`:** Responsible for filtering data and building datasets that can be passed to anomaly detectors.
- **`anomaly_detection`:** Contains implementations of anomaly detectors.
- **`postprocessing`:** Can transform datasets into lists of segments for a certain `segmentation distance`. Contains the `Segment` class, in which the `anomaly_score` function holds different definitions of the anomaly score of an IPOC segment.
- **`evaluation`:** Takes the postprocessed output of an anomaly detector and calculates its performance using a certain evaluation metric.
- **`pipeline`:** Combines all of the modules above to execute a whole anomaly detection process and report its results. Also provides grid search functionality and takes many different parameters.

The Isolation Forest and GMM anomaly detectors were implemented using the "scikit-learn" machine learning Python module [18].

Chapter 7

Conclusion

The goal of this thesis was the development of an anomaly detection application that could detect anomalous behaviour in MKI magnet data. This goal has been met; a machine learning pipeline for the detection of anomalies has been proposed. A novel evaluation procedure has been developed to diminish the uncertainty of labels and compare the performance of different models correctly.

The best performance achieved yet with the pipeline leaves something to be desired, though. The best performing parameters, which have been discussed in section 6.1, resulted in an unsatisfactory area under the PR curve of 0.423. With this performance, the pipeline would result in too many anomalies to be overlooked or too many false positives to be predicted in new scenarios. However, there is still room for improvement. Limited experiments have shown that an adapted set of features can result in significantly better performance. Also, while many parameter combinations have already been tested, there most likely are unexplored combinations that would result in better performance.

To use the pipeline for anomaly detection in new scenarios, an anomaly score threshold needs to be chosen so that segments that have an anomaly score that exceeds this threshold can be flagged as anomalous. The threshold should be chosen by CERN based on how costly it would be to overlook anomalies versus the cost of predicting false positives. First and foremost, however, the parameters of the pipeline should optimized more so that better performance can be achieved.

7.1 Future Work

Feature Set The feature set built for this thesis consisted of almost all of the collections that could be combined. It is possible that there exists a feature subset that achieves much better performance than the one that was chosen. The results have shown this to be the case for the Isolation Forest model; better results were achieved when temperature measurements were removed from the feature set. The different feature subset generation heuristics reviewed by Sutha [22] could serve as a starting point for the development of a better feature set.

Another option that could be explored in the search towards a better feature set is Controller data. Near the end of this thesis, it was noted by an expert from CERN

that in some cases, Controller data would be necessary for anomalies to be detected.

Anomaly Detectors As mentioned in section 6.3, the anomaly detection pipeline has been developed in such a way that the components can be implemented in different ways. It would be straightforward to implement more anomaly detectors using the "scikit-learn" module. For example, it could be interesting to add anomaly detectors based on the `OneClassSVM` and `LocalOutlierFactor` classes provided by "scikit-learn" and study their performance. If different detectors are good at finding different kinds of anomalies, it might be possible for an ensemble of detectors to achieve a better overall performance.

Segmentation In the postprocessing step of the pipeline, the output of an anomaly detector is transformed into a set of segments to be passed to the evaluation step. These segments represent sessions of machine usage. However, the segmentation of the data does not happen based on information about sessions that occurred in reality, but on the `segmentation distance` parameter. In this thesis, the value chosen for this parameter was the one that performed best in experiments. It would be interesting to find a segmentation process based on some ground truth that guarantees that segments correspond to machine usage sessions correctly. This way, the parameter can be removed from the pipeline and IPOC segments are truer to their definition.

Parameter Optimization Instead of grid search, another parameter optimization approach could be used so that parameters can be can be evaluated in a more intelligent and automated manner. Evolutionary algorithms and Bayesian Optimization [19] seem like good options. These approaches can test any of the possible values for parameters, instead of only those in a predetermined grid. They also automatically learn which combinations of parameters perform well, an explore more around those combinations that perform well so that even better ones can be found.

Appendix

A List of all Data Collections

LHC.BCTFR.A6R4.B1:BEAM_INTENSITY
LHC.BCTFR.A6R4.B2:BEAM_INTENSITY
LHC.BQM.B1:BUNCH_LENGTH_MEAN
LHC.BQM.B2:BUNCH_LENGTH_MEAN
MKI.A5L2.B1:PRESSURE
MKI.A5L2.B1:TEMP_MAGNET_DOWN
MKI.A5L2.B1:TEMP_MAGNET_UP
MKI.A5L2.B1:TEMP_TUBE_DOWN
MKI.A5L2.B1:TEMP_TUBE_UP
MKI.A5R8.B2:PRESSURE
MKI.A5R8.B2:TEMP_MAGNET_DOWN
MKI.A5R8.B2:TEMP_MAGNET_UP
MKI.A5R8.B2:TEMP_TUBE_DOWN
MKI.A5R8.B2:TEMP_TUBE_UP
MKI.B5L2.B1:PRESSURE
MKI.B5L2.B1:TEMP_MAGNET_DOWN
MKI.B5L2.B1:TEMP_MAGNET_UP
MKI.B5L2.B1:TEMP_TUBE_DOWN
MKI.B5L2.B1:TEMP_TUBE_UP
MKI.B5R8.B2:PRESSURE
MKI.B5R8.B2:TEMP_MAGNET_DOWN
MKI.B5R8.B2:TEMP_MAGNET_UP
MKI.B5R8.B2:TEMP_TUBE_DOWN
MKI.B5R8.B2:TEMP_TUBE_UP
MKI.C5L2.B1:PRESSURE
MKI.C5L2.B1:TEMP_MAGNET_DOWN
MKI.C5L2.B1:TEMP_MAGNET_UP
MKI.C5L2.B1:TEMP_TUBE_DOWN
MKI.C5L2.B1:TEMP_TUBE_UP
MKI.C5R8.B2:PRESSURE
MKI.C5R8.B2:TEMP_MAGNET_DOWN
MKI.C5R8.B2:TEMP_MAGNET_UP
MKI.C5R8.B2:TEMP_TUBE_DOWN
MKI.C5R8.B2:TEMP_TUBE_UP
MKI.D5L2.B1:PRESSURE
MKI.D5L2.B1:TEMP_MAGNET_DOWN
MKI.D5L2.B1:TEMP_MAGNET_UP
MKI.D5L2.B1:TEMP_TUBE_DOWN
MKI.D5L2.B1:TEMP_TUBE_UP
MKI.D5R8.B2:PRESSURE
MKI.D5R8.B2:TEMP_MAGNET_DOWN
MKI.D5R8.B2:TEMP_MAGNET_UP
MKI.D5R8.B2:TEMP_TUBE_DOWN
MKI.D5R8.B2:TEMP_TUBE_UP
MKI.ELOGBOOK
MKI.ELOGBOOK_tagged
MKI.UA23.F3.CONTROLLER:
KICK_COUNT_TOPLAY
MKI.UA23.F3.CONTROLLER:
KICK_DELAY_TOPLAY
MKI.UA23.F3.CONTROLLER:
KICK_ENABLE_TOPLAY
MKI.UA23.F3.CONTROLLER:
KICK_LENGTH_TOPLAY
MKI.UA23.F3.CONTROLLER:
KICK_STRENGTH_TOPLAY
MKI.UA23.F3.CONTROLLER:
KICK_TIME_TOPLAY
MKI.UA23.IPOC.AB1:E_KICK
MKI.UA23.IPOC.AB1:I_STRENGTH
MKI.UA23.IPOC.AB1:T_DELAY
MKI.UA23.IPOC.AB1:T_FALLTIME

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

Appendix

B Labels

A few labels as mentioned in subsection 4.1.6.

C0	9
EVENTDATE	02/05/2016 14:59:00
EVENT_ID	2104506
PATH	LHC.MKI8
COMMENT	Stopped the MKI8 today around 12.00 am to 'burn' the EPROM to ensure the latest modifications (specifically to the INJECTION_PERMIT) are now inside the PLC. Put SYSTEM back in STANDBY (with correct masks applied where required). ALL good.
TAG	intervention
VALUE	MKI8
C0	10
EVENTDATE	03/05/2016 08:30:00
EVENT_ID	2104890
PATH	LHC.MKI8
COMMENT	Increased SIS vacuum integral threshold from 1e-3 to 2e-3 for only magnets MKI8A and MKI8D. This is to compensate for the Penning gauge, on these 2 magnets, having a relatively high reading due to the heating collar on these gauges. All other MKIs have a vacuum integral threshold of 1.0e-3.
TAG	info
VALUE	MKI8
C0	61
EVENTDATE	28/07/2016 15:04:31
EVENT_ID	2151999
PATH	LHC.MKI2
COMMENT	Attached screen shot is from scope for MKI2. Yellow trace is TMR for MKI2D Cyan and Purple traces are CPUin and CPUout, respectively, for MKI2D Green trace is TDR for MKI2D Delay is changed to examine maximum length of TMR pulse (for MD). Pulse lengths shown are: 4200ns, 5200ns, 6200ns, 7200ns, 8200ns, 8700ns, 8900ns, 9000ns
TAG	research
VALUE	MKI2

C0	66
EVENTDATE	02/08/2016 19:50:00
EVENT_ID	2155328
PATH	LHC.MKI2
COMMENT	24 strong sparks were detected during the Extended Softstart. The automatic conditioning stopped at 50kV due to those sparks detection (vacuum increased to 1.1e-8). Still no spark in IPOC. Viliam called me back and asked me to call Mike B. Mike checked the data and asked a Spark reset and Extended Softstart. I called CCC back at 20:51 and requested an Extended Softstart. Left CERN at 21:10.
TAG	anomaly
VALUE	MKI2
C0	87
EVENTDATE	21/08/2016 22:00:00
EVENT_ID	2165241
PATH	LHC.MKI8
COMMENT	20.20. MKI8 Beam 2 Magnet B. Vacuum spike. I was still at the PS with Etienne when Etienne was called by the LHC CCC.I called Laurent to take a look at the problem but no answer. I called Viliam and he kindly tried to reset the fault but as it was not actually a Vacuum interlock he found that he did not have the rights to reset it...it was Just a spark and this created a FSC Magnet fault. This is a Piquet role only reset it seems..... He was given an RBAC token in anycase but he could reset as he still did
TAG	anomaly
VALUE	MKI8
C0	91
EVENTDATE	30/08/2016 06:00:30
EVENT_ID	2171093
PATH	LHC.MKI2
COMMENT	CCC-LHC operator calls for the MKI2 BEAM 1 is faulty during the injection sequence of LHC. In Remote : The MKI2 is faulty with : - Generator 1 Fault on the step Capacitor Bank Switch with Resonant Charging fault detected - Generator 2 Fault on the step 5-Capacitor Bank Switch with Resonant Charging fault detected and step 6-PFN switches activated with the fault FSC Warning. Erratic has been detected on the MAIN switch of PFN-D Counter was the Value 1 - Counter has been reserted No Vacuum fault detected
TAG	fault
VALUE	MKI2

Appendix

C Grid Search Output

Excerpt from the output of a grid search run (subsection 5.4.3) using GMM. Results are collected at the end and listed in descending order of score.

```
| Training GMM with params: {'n_components': 8, 'verbose': 1, '
    covariance_type': 'full', 'n_init': 1, 'init_params': 'kmeans'},
    scale_data = False
|       Training: 74.1s
| Evaluation params: segmentation_distance = 15, anomaly_score_method = max,
     types_of_labels = anomaly
|       AUPR = 0.3948 in 7.3s (193/288)
| Evaluation params: segmentation_distance = 30, anomaly_score_method = max,
     types_of_labels = anomaly
|       AUPR = 0.3953 in 5.6s (194/288)
| Evaluation params: segmentation_distance = 15, anomaly_score_method = max,
     types_of_labels = any
|       AUPR = 0.2512 in 16.8s (195/288)
| Evaluation params: segmentation_distance = 30, anomaly_score_method = max,
     types_of_labels = any
|       AUPR = 0.2539 in 13.3s (196/288)
| Evaluation params: segmentation_distance = 15, anomaly_score_method =
    top_k, types_of_labels = anomaly
|       AUPR = 0.4008 in 8.4s (197/288)
| Evaluation params: segmentation_distance = 30, anomaly_score_method =
    top_k, types_of_labels = anomaly
|       AUPR = 0.4031 in 6.4s (198/288)
| Evaluation params: segmentation_distance = 15, anomaly_score_method =
    top_k, types_of_labels = any
|       AUPR = 0.2516 in 18.1s (199/288)
| Evaluation params: segmentation_distance = 30, anomaly_score_method =
    top_k, types_of_labels = any
|       AUPR = 0.2546 in 13.9s (200/288)
| Evaluation params: segmentation_distance = 15, anomaly_score_method =
    top_percentage, types_of_labels = anomaly
|       AUPR = 0.1655 in 8.6s (201/288)
| Evaluation params: segmentation_distance = 30, anomaly_score_method =
    top_percentage, types_of_labels = anomaly
```

```
|      AUPR = 0.1702 in 6.3s (202/288)
| Evaluation params: segmentation_distance = 15, anomaly_score_method =
    top_percentage, types_of_labels = any
|      AUPR = 0.1482 in 17.9s (203/288)
| Evaluation params: segmentation_distance = 30, anomaly_score_method =
    top_percentage, types_of_labels = any
|      AUPR = 0.1568 in 13.9s (204/288)

RESULTS:

| AUPR = 0.403 in 80.5s for seg_distance = 30, a_score_method =     top_k,
    types_of_labels = anomaly, params = {'n_components': 8, 'verbose': 1, '
    covariance_type': 'full', 'n_init': 1, 'init_params': 'kmeans'}
| AUPR = 0.401 in 82.6s for seg_distance = 15, a_score_method =     top_k,
    types_of_labels = anomaly, params = {'n_components': 8, 'verbose': 1, '
    covariance_type': 'full', 'n_init': 1, 'init_params': 'kmeans'}
| AUPR = 0.395 in 79.7s for seg_distance = 30, a_score_method =       max,
    types_of_labels = anomaly, params = {'n_components': 8, 'verbose': 1, '
    covariance_type': 'full', 'n_init': 1, 'init_params': 'kmeans'}
| AUPR = 0.395 in 81.4s for seg_distance = 15, a_score_method =       max,
    types_of_labels = anomaly, params = {'n_components': 8, 'verbose': 1, '
    covariance_type': 'full', 'n_init': 1, 'init_params': 'kmeans'}
| AUPR = 0.255 in 88.0s for seg_distance = 30, a_score_method =     top_k,
    types_of_labels =  any, params = {'n_components': 8, 'verbose': 1, '
    covariance_type': 'full', 'n_init': 1, 'init_params': 'kmeans'}
| AUPR = 0.254 in 87.4s for seg_distance = 30, a_score_method =       max,
    types_of_labels =  any, params = {'n_components': 8, 'verbose': 1, '
    covariance_type': 'full', 'n_init': 1, 'init_params': 'kmeans'}
| AUPR = 0.252 in 92.2s for seg_distance = 15, a_score_method =     top_k,
    types_of_labels =  any, params = {'n_components': 8, 'verbose': 1, '
    covariance_type': 'full', 'n_init': 1, 'init_params': 'kmeans'}
| AUPR = 0.251 in 90.9s for seg_distance = 15, a_score_method =       max,
    types_of_labels =  any, params = {'n_components': 8, 'verbose': 1, '
    covariance_type': 'full', 'n_init': 1, 'init_params': 'kmeans'}
| AUPR = 0.170 in 80.4s for seg_distance = 30, a_score_method =
    top_percentage, types_of_labels = anomaly, params = {'n_components': 8,
    'verbose': 1, 'covariance_type': 'full', 'n_init': 1, 'init_params': '
    kmeans'}
| AUPR = 0.165 in 82.7s for seg_distance = 15, a_score_method =
    top_percentage, types_of_labels = anomaly, params = {'n_components': 8,
    'verbose': 1, 'covariance_type': 'full', 'n_init': 1, 'init_params': '
    kmeans'}
| AUPR = 0.157 in 88.0s for seg_distance = 30, a_score_method =
    top_percentage, types_of_labels = any, params = {'n_components': 8, '
    verbose': 1, 'covariance_type': 'full', 'n_init': 1, 'init_params': '
    kmeans'}
| AUPR = 0.148 in 92.0s for seg_distance = 15, a_score_method =
    top_percentage, types_of_labels = any, params = {'n_components': 8, '
    verbose': 1, 'covariance_type': 'full', 'n_init': 1, 'init_params': '
    kmeans'}
```

Appendix

D AnomalyDetector interface

The source code for this thesis (section 6.3) was written in Python. The listing below shows the most important functions of the `AnomalyDetector` interface in the `anomaly_detection` package.

```
1  from abc import ABC, abstractmethod
2
3  class AnomalyDetector(ABC):
4      """ Base anomaly detector class """
5
6      @abstractmethod
7      def set_params(self, **params):
8          """ Sets the parameters of the detector. """
9          raise NotImplementedError
10
11     @abstractmethod
12     def fit(self, df):
13         """ Trains the detector on the given DataFrame. """
14         raise NotImplementedError
15
16     @abstractmethod
17     def anomaly_scores(self, df):
18         """ Given a DataFrame of features, returns anomaly scores for all samples.
19         Anomaly scores lie in [0, 1] and larger scores are more anomalous.
20         """
21         raise NotImplementedError
22
23     @abstractmethod
24     def predict(self, df):
25         """ Given a DataFrame of features, returns class predictions all samples. """
26         raise NotImplementedError
27
28
```

```
29      def fit_and_anomaly_scores(self, df):
30          """ Fits the model on the given DataFrame and scores each sample. """
31          self.fit(df)
32          return self.anomaly_scores(df)
33
34      def fit_and_append_scores(self, df):
35          """ Trains the detector on the given DataFrame, scores it, and appends the
                scores to it. """
36          scores = self.fit_and_anomaly_scores(df)
37          return df.assign(score=scores)
```

Appendix

E Poster

Anomaly detection for the Large Hadron Collider

KATHOLIEKE UNIVERSITEIT LEUVEN

FACULTEIT INGENIEURSWETENSCHAPPEN

Master of Engineering: Computer Science

Student:
Armin Halilovic

Promotors:
Wannes Meert
Hendrik Blockeel

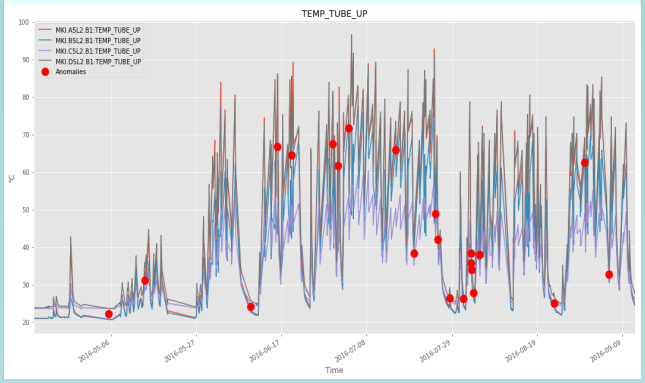
Advisor:
Elia van Wolputte

External advisor:
Pieter van Trappen

Academic year 2017-2018

Context

- LHC created for particle physics research
- **Injection Kicker Magnets (MKI)** inject counter-rotating proton beams into the LHC at an energy of 450 GeV
- 2 MKI installations of 4 magnets
- 120 datasets: pressure, temperatures, kick strength, current, risetime, falltime, controller state, ...



Problem

- Anomalous behavior can cause experiments to fail => time/money waste
- Manual analysis after fault occurence is very time-consuming

Goal

Develop an anomaly detection application that can automatically detect anomalous behavior from MKI data

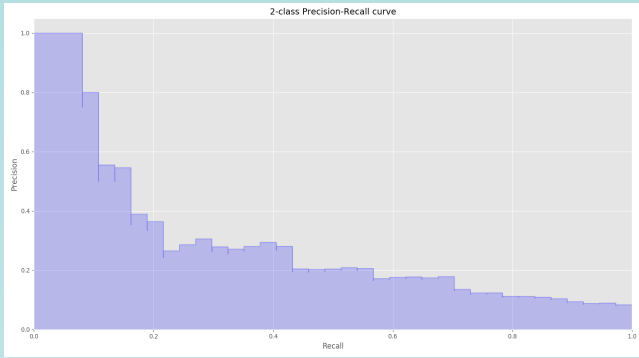
Challenges

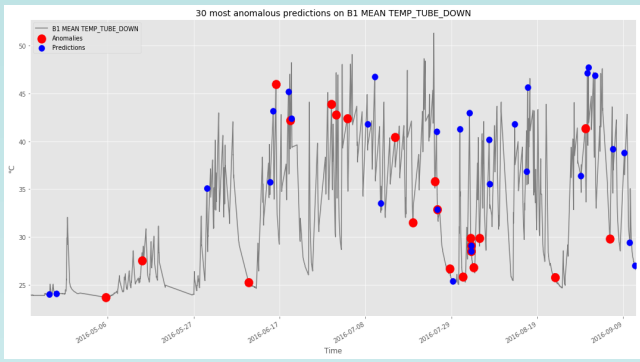
- No consistent “correct” behavior
- Different types of anomalous behavior
- Mixed data sampling rates and times
- Small set of labeled data
- Labels are not precise

Approach

- **Preprocessing**
  - Filter incorrect measurements
- **Feature extraction**
  - Only data when magnets were in use
  - Temporal features (moving average/sum)
- **Anomaly detection**
  - Unsupervised
  - Isolation Forest
  - Explicitly isolates anomalies
  - Anomaly score for every timepoint
  - Scores in [-1, 1]
- **Evaluation**
  - Group data into “segments” that represent periods of magnet usage
  - Calculate anomaly scores for segments
  - Threshold on scores for classification
  - Use labels to mark which segments are actually anomalous
  - Compare results using **area under precision-recall curve**

Results





Conclusion

- Low precision and recall means that the current application is not usable
- Either a lot of false positives, or low detection of anomalies
- Future work: better feature extraction

Bibliography

[1] R. A. Barlow, M. Barnes, E. Carlier, and M. Laffin. Kiss conditioning of the lhc injection kickers. March 2010.

[2] M. J. Barnes, L. Ducimetière, T. Fowler, V. Senaj, and L. Sermeus. Injection and extraction magnets: kicker magnets. (arXiv:1103.1583):1–26, March 2011. Presented at the CERN Accelerator School CAS 2009: Specialised Course on Magnets, Bruges, 16-25 June 2009.

[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, November 2006.

[4] H. Blockeel. *Machine Learning and Inductive Inference*. Acco, 5th edition, 2010. ISBN-13: 978-90-334-8297-7.

[5] O. Brüning, P. Collier, P. Lebrun, S. Myers, R. Ostojic, J. Poole, and P. Proudlock. *LHC Design Report, v.1: The LHC Main Ring*. EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH, 2004. ISBN: 92-9083-224-0.

[6] CERN. The accelerator complex. https://cds.cern.ch/record/1997193, January 2012. Accessed 2017-12-25.

[7] CERN. Pulling together: Superconducting electromagnets. http://cds.cern.ch/record/1997395, August 2012. Accessed 2017-12-25.

[8] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3), 2009.

[9] J. Davis and M. Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM, 2006.

[10] J. L. Devore. *Probability and Statistics for Engineering and the Sciences*. Brooks/Cole, 8th edition, January 2011. ISBN-13: 978-0-538-73352-6.

[11] L. Ducimetière, N. Garrel, M. J. Barnes, and G. D. Wait. The lhc injection kicker magnet. (LHC-Project-Report-655. CERN-LHC-Project-Report-655):1–4, June 2003.

[12] R. T. G. for the LHC team. Lhc report: machine development. (BUL-NA-2015-169. 32/2015):1–3, July 2015.

[13] W. Herr and B. Muratori. Concept of luminosity. 2006.

[14] W. Herr and T. Pieloni. Beam-beam effects. (arXiv:1601.05235):1–29, 2014. Contribution to the CAS - CERN Accelerator School: Advanced Accelerator Physics Course, Trondheim, Norway, 18-29 Aug 2013.

[15] K. Javed, H. A. Babri, and M. Saeed. Feature selection based on class-dependent densities for high-dimensional binary data. *IEEE Transactions on Knowledge and Data Engineering*, 24(3):465–477, March 2012.

[16] F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation forest. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 413–422. IEEE, 2008.

[17] E. Mobs. The cern accelerator complex. http://cds.cern.ch/record/2197559/files/?docname=CCC-v2017&version=all, 2017. Accessed 2018-06-04.

[18] 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.

[19] M. Pelikan, D. E. Goldberg, and E. Cant ú Paz. Boa: The bayesian optimization algorithm. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 1*, pages 525–532. Morgan Kaufmann Publishers Inc., 1999.

[20] M. A. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko. A review of novelty detection. *Signal Processing*, 99:215–249, 2014.

[21] B. Puccio, A. Castañeda Serra, M. Kwiatkowski, I. Romera Ramirez, and B. Todd. The cern beam interlock system: Principle and operational experience. (CERN-ATS-2010-128):4 p, June 2010.

[22] K. Sutha. A review of feature selection algorithms for data mining techniques. 2015.

[23] J. R. Wells, K. M. Ting, and N. P. Chandrasiri. A non-time series approach to vehicle related time series problems. In *AusDM*, 2012.

[24] N. Wéry. Anomalie detectie voor de large hadron collider. Master's thesis, KU Leuven, 2017.

Master's thesis filing card

Student: Armin Halilovic

Title: Anomaly Detection for the CERN Large Hadron Collider injection magnets

Dutch title: Anomalie detectie voor de CERN Large Hadron Collider injectie magneten

UDC: 681.3*I20

Abstract:

The Large Hadron Collider is the world's largest single machine. Inside it, high-energy particle beams are made to collide at speeds near the speed of light. The results of the collisions are then analysed in the hope that some of the fundamental open questions in physics can be answered. Particle beams are injected into the LHC and guided throughout it by thousands of electromagnets. New state of the art equipment was developed to make all of this possible. The result of this is that, due to yet undiscovered reasons, equipment occasionally behaves in unexpected ways. This behaviour is called anomalous behaviour and it can cause experiments to fail. Experiments must then be restarted, which is a costly procedure. The goal of this thesis is to develop a method that can detect anomalies in the behaviour of injection kicker magnets, the magnets that inject particles into the LHC.

A system that detects anomalies cannot be built manually, as there are over a hundred data signals related to the magnets and many complex relationships exist between the signals. Currently, anomalies are only detected at CERN after they have caused experiments to fail. A machine learning based application that can learn the complex relationships between signals and that can then detect anomalies automatically would thus greatly benefit CERN. Reaction times to anomalies could be improved and less time would need to be spent meticulously analysing signal data to find the causes of anomalies. Furthermore, it is possible for machine learning models to list the factors that contribute most to anomalous behaviour. Such information could be used by CERN to improve equipment.

This thesis proposes an extensible anomaly detection application which supports the use of many different machine learning models. There is a limited amount of unspecific knowledge about when and where anomalies occurred in injection kicker magnet behaviour. A novel evaluation procedure is introduced that can use this unspecific knowledge to measure and compare anomaly detection performance fairly.

Thesis submitted for the degree of Master of Science in Engineering: Computer Science, option Artificial Intelligence

Thesis supervisors: Prof. Dr. ir. Hendrik Blockeel
Dr. ir. Wannes Meert

Assessors: Prof. Dr. Bruno Crispo
Dr. Stefano Teso

Mentors: Elia Van Wolputte
Ir. Pieter Van Trappen