ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEMPR010

ANOMALY DETECTION FOR CERN BEAM TRANSFER INSTALLATIONS USING MACHINE LEARNING

T. Dewitte, W. Meert, E. Van Wolputte[*], KU Leuven, Leuven, Belgium
P. Van Trappen, CERN, Geneva, Switzerland

Abstract

Reliability, availability and maintainability determine whether or not a large-scale accelerator system can be operated in a sustainable, cost-effective manner. Beam transfer equipment (e.g. kicker magnets) has potentially significant impact on the global performance of a machine complex. Identifying root causes of malfunctions is currently tedious, and will become infeasible in future systems due to increasing complexity. Machine Learning could automate this process. For this purpose a collaboration between CERN and KU Leuven was established.

We present an anomaly detection pipeline which includes preprocessing, detection, postprocessing and evaluation. Merging data of different, asynchronous sources is one of the main challenges. Currently, Gaussian Mixture Models and Isolation Forests are used as unsupervised detectors. To validate, we compare to manual e-logbook entries, which constitute a noisy ground truth. A grid search allows for hyper-parameter optimization across the entire pipeline. Lastly, we incorporate expert knowledge by means of semi-supervised clustering with COBRAS.

INTRODUCTION

The CERN Large Hadron Collider (LHC) is the world's largest and most powerful particle accelerator [1]. Maintaining a machine of this size in an efficient and sustainable manner is highly non-trivial. Currently, detection of problematic situations (anomalies) during operation mostly happens manually; experts analyze the data, identify anomalies and track down root causes. In the case of equipment failure, this manual process results in the loss of precious machine time. Furthermore, this manual approach requires strong knowledge of the complex relations between LHC components and does not scale to future – larger – accelerators.

We present a proof-of-concept, scalable solution for automatic anomaly detection in the LHC. Concretely, Machine Learning (ML) methods model normal behaviour of its injection kicker magnets, based on historical data. Afterwards, this learned model flags unexpected behaviour in unseen data. We situate our work, describe the data, explain the design of the application and finally report on our experiments, including one on incorporating expert feedback into the ML system.

[*] Supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme, grant agreement No. 694980 "SYNTH: Synthesising Inductive Data Models"

INJECTION KICKER MAGNETS

Within the CERN accelerator complex, particle beams are extracted from one accelerator and injected into another by means of beam transfer equipment. At CERN it is the responsibility of the Accelerator Beam Transfer (TE/ABT) group to design, install and maintain this equipment. Especially critical for LHC operation are the Injection Kicker Magnets (MKI) [2], a set of two times four magnets that inject beams from the SPS into the LHC.

An MKI installation consists of four magnets, which are named A, B, C and D. Each pair of magnets (A-B and C-D) is powered by one generator, with a single high-voltage resonant charging power supply (RCPS) to charge two pulse forming networks (PFNs); thus one PFN per magnet. The LHC requires two injection installations (named MKI2 and MKI8), one for each counter-rotating beam. This brings the total of magnets to eight.

DATA

Typically, ML algorithms assume clean datasets of tabular form: each row represents a single observation, each column represents a feature. However, the raw data collected by CERN does not obey this ideal form; distilling sensible feature vectors from such a complex mixture of (mostly) asynchronous data sources constitutes a key challenge. This complexity stems mostly from the fact that the data sources were never set up with downstream ML applications in mind. Concretely, the dataset contains time series data (continuous), IPOC data (event data), state data (categorical), controller data (numerical), general LHC data (numerical), and logbook data (categorical and text).

Continuous, Time Series Data

First, we have continuous, time series data, e.g. pressure and temperature measurements of the MKI installation. The temperature measurements occur at two points, where particles enter the magnet (*upstream*) and where they exit the magnet (*downstream*). This continuous data is sampled at a fixed frequency of 2 Hz. To increase logging efficiency, values are often only stored upon change, or after a certain threshold duration without any change [3]. This threshold is configured by the CERN user and ranges from 15 minutes to one day. For each magnet, five such continuous variables are measured. Given four magnets (A, B, C, D) per system (MKI2, MKI8), this yields 40 continuous variables in total.

Internal Post Operational Check Data

The second type of data is so-called Internal Post Operational Check (IPOC) data [4]. Examples include kick strength and length of the magnet current pulse.

Unlike the continuous data, IPOC data is not sampled at a fixed frequency. Instead, collection relies on *acquisition trigger sampling* meaning that when a certain trigger occurs, hardware digitizers capture data from their sensors at that instant[1]. Such a trigger corresponds to usage of the magnets (i.e. fast pulsed current to deflect the beam into the LHC), making these timestamps useful in understanding the behaviour of the machine.

Just like the time series data, IPOC variables are associated with individual magnets. Given seven IPOC variables/magnet, the dataset contains 56 IPOC variables.

PLC State Data

The third type of data originates from a high-level surveying Programmable Logic Controller (PLC) and records the state of the MKI magnets. This is a time series of categorical data. Two state variables are of particular importance. First, `STATE:CONTROL` indicates whether the control room has control over the magnets (*remote*), or whether it was used for research or testing by the equipment group (*local*). Second, `STATE:MODE` contains information about actual machine usage; whenever this changes from `ON` to any other value, it indicates a finished injection cycle. Per MKI installation (MKI2, MKI8), four state data variables are recorded, yielding eight state variables in total.

Timing Controller Data

The fourth type of data are timing-related operational parameters of the MKI magnets, to determine for instance the amplitude and length of a pulse. This is event-based, numerical data. Like the state data, the controller data (six variables) is also associated with an entire MKI installation, resulting in 12 controller variables.

LHC Data

The fifth type of data are (numerical) measurements of beam intensity and bunch length in the LHC. This is continuous, time series data. Albeit not specific to the MKI magnets, there are some obvious correlations, e.g. higher beam intensity leads to increased temperatures. Both LHC variables are stored for beam 1 and 2, which results in four LHC variables.

Electronic Logbook

Lastly, our dataset contains an annotated, electronic logbook (*e-logbook*). This e-logbook consists of a collection of entries, made by CERN staff when the protocol requires to do so. Additionally, this e-logbook was analyzed and annotated by a CERN expert with one of the following labels: `anomaly`, `fault`, `info`, `intervention` or `research`. Entries labelled `anomaly` indicate machine behaviour that our application should detect automatically. Two remarks are in place. First, the e-logbook is prone to human error. Some events may be added late, or, if undiscovered, not at all. In consultation with CERN experts, we associate an uncertainty range of 12 hours to timestamps of e-logbook entries. Second, as is common in anomaly detection, we have far less labels than we have data. This excludes supervised ML methods as anomaly detectors.

[1] Importantly, IPOC data represent results from voltage or current waveform analysis; an industrial PC runs an analysis algorithm and the properties of the waveform are extracted as IPOC data.

Summary

On the one hand, we have a complex dataset, encompassing 120 variables coming from a variety of data sources. This represents the machine behaviour we wish to model. On the other hand, we have the e-logbook, and specifically its entries labelled `anomaly`. It serves as noisy ground truth by which we measure the success of our anomaly detection application.

ANOMALY DETECTION APPLICATION

Essentially, the anomaly detection application (Fig. 1) is structured as a data pipeline of five consecutive steps: preprocessing, anomaly detection, postprocessing, evaluation and visualisation.

Preprocessing

The first step transforms raw data into a set of clean feature vectors suitable for ML. This encompasses filtering, scaling, feature engineering and finally data fusion.

Filtering is the removal of unwanted data. First, we ignore erroneous measurements, e.g. a negative delay value. Second, we also omit data generated during machine interventions, as these do not constitute normal behaviour.

Scaling normalizes the input data: the mean is set to zero, the standard deviation to one. The user can optionally disable this step.

Feature engineering refers to the explicit creation of new features, alongside those extracted directly from the dataset. The transition of raw data to clean feature vectors will not preserve the temporal nature of our dataset. Since temporal effects may well be relevant for anomaly detection, we incorporate temporal information in the feature vectors via custom-made features. Concretely, we introduce sliding window and Fourier features.

Data fusion refers to the actual merging of many raw data sources into an “ML-ready” feature vector representation. We start from the event-based IPOC data: each IPOC measurement corresponds to an instance of actual machine usage, represented by a 56-dimensional feature vector. The main idea is to add additional data to these IPOC feature vectors by extrapolation of non-IPOC data to IPOC timestamps. Two remarks are in place. First, magnet usage only

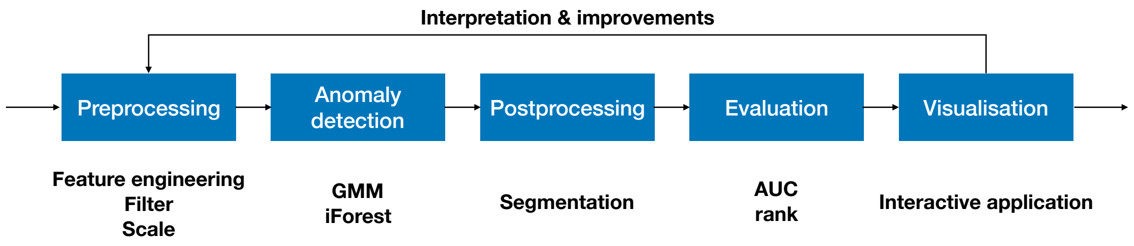

Figure 1: The five-step anomaly detection pipeline: preprocessing, anomaly detection, postprocessing, evaluation and visualisation. Concrete actions are mentioned under their respective sections.

accounts for a tiny fraction of the dataset; we assume that anomaly detection in MKI magnets only requires usage data of such magnets (note however that beam-induced magnet heating and pressure drops are thus ignored). Second, IPOC timestamps are rounded up to the second: this limits the amount of new data generated during resampling of other data sources and does not entail any information loss (there is at most one IPOC value per second). Next, all non-IPOC data is extrapolated to these IPOC timestamps and added to their corresponding feature vectors. Ultimately, this results in a set of feature vectors, one per IPOC-timestamp, containing IPOC, state, controller, LHC and continuous data. This also includes sliding window features for both continuous and LHC data, and Fourier features for the continuous data.

Anomaly Detection

In the second step, an *unsupervised* ML model (i.e. the anomaly detector) is trained on the feature vectors. Here, unsupervised means unlabelled: the ML model does not know which feature vectors represent anomalous behaviour. Given that anomalies only constitute a small subset of the entire dataset, the ML algorithm can still learn normal behaviour. After training, this detector quantifies how *anomalous* a given feature vector is, according to its internal model of normal behaviour.

Our framework supports any off-the-shelf, unsupervised, anomaly detector which takes a feature vector as input and yields an anomaly score $a \in [0, 1]$ as output. In experiments, we focus on Gaussian Mixture Models (`GMM`) and Isolation Forests (`iForest`).

ML algorithms come with a set of so-called *hyper-parameters*, i.e. parameters set by the user. E.g. in `GMM`, the number of Gaussian distributions is a hyper-parameter. Finding good values for these hyper-parameters is done automatically through a grid search.

Postprocessing

The third step assigns each scored feature vector to a segment, i.e. a time interval which corresponds to a period of MKI-activity. Each segment has its own aggregate anomaly score[2] and ground truth. The reason for this segmentation is twofold. First, anomaly detection at the level of individual timestamps is too fine-grained for CERN. The question is whether or not the MKI magnets work as expected, not whether or not one data-point seems a bit off. Second, due to the fuzzy nature of the e-logbook (our only source of ground truth) we cannot reliably assign ground truth labels to individual timestamps. We can do so at the level of individual segments.

Segment boundaries are based on the `STATE:MODE` data; a change of this variable to `ON` marks a segment start, a change away from `ON`, a stop. To ensure a correct evaluation afterwards, we assign a ground truth to each segment, and occasionally we merge multiple segments into one.

Segment ground truth For evaluation purposes, each segment needs a *ground truth*. Here, the labelled e-logbook entries come into play. However, these labels are noisy: the actual anomaly might have occurred some hours before or after[3] the timestamp of the e-logbook entry. To account for this uncertainty, a segment is considered anomalous if it occurs within a 12h-range of an e-logbook entry labelled `anomaly`.

Segment merging Occasionally, the uncertainty range of the e-logbook introduces some ambiguities, which are resolved by merging several segments. For instance: consider one anomalous e-logbook entry at 3 PM and two segments, one from 1-2 PM and another from 4-5 PM. Now, our methodology assigns an anomalous ground truth to both segments, which requires our anomaly detector to detect this label twice. To ensure that each anomalous label of the e-logbook has to be detected once, it suffices to merge all corresponding segments into one, prior to evaluation.

Evaluation

Fourth, we compare the ground truth with the outcomes of our anomaly detector, and quantify its performance by means of two evaluation metrics: AUC and rank.

First, the area under the precision-recall curve (AUC). This curve is obtained by computing precision and recall for a varying decision threshold. The maximum AUC is 1, and the higher the better.

[2] The anomaly scores of all feature vectors in the segment are combined to yield the segment anomaly score. The precise manner of aggregation (e.g. mean of top-k entries) is a user-specified parameter.

[3] CERN staff can manually assign a timestamp to an e-logbook entry, so it can be early or late with respect to the event it describes.

Second, average anomaly rank (rank). We sort all segments based on their anomaly scores a assigned by the detector. Suppose n truly anomalous segments, a perfect detector isolates these at the top of its ranking, yielding a minimum average anomaly rank of $n(n+1)/2$. So, a lower rank means a better detector.

Visualisation

Finally, we introduce an interactive explorer, available online[4]. It allows to inspect both the data variables, e-logbook and predictions with live metrics.

EXPERIMENTS

Hyper-Parameter Grid Search

The first experiment automatically optimizes the hyper-parameters of the ML model, through a grid search. Such hyper-parameters (e.g. the number of Gaussian distributions used in `GMM`) strongly influence the performance of a ML model, so pinpointing sensible values matters.

Set-up Given a search space of potential hyper-parameter values (i.e. the grid), we run the entire pipeline for each combination, and we evaluate according to AUC.

Results Finally, we obtain a set of optimal hyper-parameters for each anomaly detector per year. Based on these results and efficiency considerations, we propose a general set of hyper-parameters for future use. Moreover, such a general set of parameters is less prone to overfitting. For `GMM`, we tested 30 configurations, shown together with the optimal result for 2018 in Table 1. These optimal parameters yield the predictions of Table 2. Note that this setup is likely to yield overly optimistic results, in the third experiment we address this issue.

Table 1: Optimal parameters for 2018 after grid search and a general proposal for decent results regardless of year.

Parameter	**Tested**	**2018**	**general**
n_components	1, 2, 4, 6, 8	2	2
covariance_type	full, tied, diag	full	full
n_init	1, 5	5	1
init_params	kmeans	kmeans	kmeans

Table 2: Predictions after grid search with `GMM` for 2018

	anomaly	normal
detected	4	11
undetected	3	1428

Feature Selection

In the second experiment, the aim is to reduce the total number of features, whilst maintaining performance. The goal is mainly to lower computational complexity, and possibly to simplify root-cause analysis.

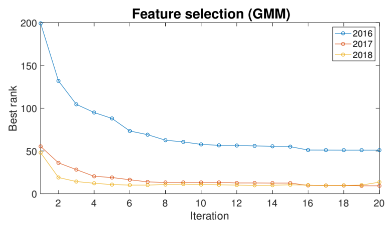

Figure 2: The progress of feature selection with `GMM`. Overall, 10 to 15 iterations are sufficient. Also remarkable is the relative difficulty of 2016 compared to the other years.

Set-up Our feature selection is structured as a greedy local search algorithm. Start with an initial solution (fixed set of features) and determine a set of neighbouring solutions (a neighbour is obtained by adding or deleting a feature from the current solution). For every neighbour, run the entire pipeline and evaluate according to its rank. Select the neighbour with the lowest rank as new solution. Repeat until you reach a maximum number of iterations. Finally, we return the solution of the last iteration (Fig. 2).

Results Using at most 16 features for `GMM` and 26 features for `iForest`, a model trained using the general parameters detects the same amount of anomalies as a model without feature selection. Smaller feature sets reduce both training time, as well as time spent interpreting detector outcomes.

Table 3: Predictions after feature selection (16 features) with `GMM` for 2018. For original performance, cf. Table 2.

	anomaly	normal
detected	5	10
undetected	2	1429

Live Evaluation

The third experiment tests our anomaly detection application in a more complex and realistic setting, designed to mimic deployment conditions. In particular, instead of having access to the complete set of historical data, data now comes in sequentially and anomalies have to be detected in previously unseen data.

Set-up Concretely, we iterate through our data in one-week increments. First, we train the anomaly detector on week 1 of historical data and detect anomalies in week 2. Its performance is recorded. Next, we train an anomaly detector on data of weeks 1 and 2, and use that model to detect anomalies in week 3. This process repeats until an entire year's worth of data is processed.

Results The results for 2018 (Table 4) show that recall remains steady (0.57), but precision suffers. However, for

[4] `http://cern.ch/anomaly-detection-mki`

Table 4: Predictions after live evaluation with GMM for 2018. Recall remains unaltered, but precision suffers due to an increased amount of false positives (FPs).

	anomaly	normal
detected	4	32
undetected	3	1376

2016 and 2017, recall does decline: some anomalies are no longer discovered. Lowering the detection threshold mitigates this effect, but introduces even more false positives (FPs). In conclusion, our detector does work during live evaluation, but one must somehow cope with a comparatively large number of FPs.

COBRAS: Incorporating Expert Feedback

To reduce the amount of FPs, our final experiment investigates how to incorporate expert feedback. In particular, we propose an additional step which interactively clusters the output of the anomaly detector using the `COBRAS` system [5, 6]. Ultimately, `COBRAS` should learn (both from data and from expert feedback) to correct mistakes of the anomaly detector by re-assigning wrongly classified segments to the right clusters of anomalous or normal segments. This knowledge can be stored and re-used on unseen data. In this way, we introduce expert feedback independent of a particular choice of anomaly detector.

COBRAS `COBRAS` clusters a given dataset such that the result aligns with the preferences of a particular user. To do so, it asks the user questions of the form: *"Do these two instances belong to the same cluster?"* Based on those constraints, `COBRAS` further refines its clustering.

Input data The segments, along with their associated anomaly scores, serve as input data to the `COBRAS` algorithm. However, `COBRAS` requires fixed-length feature vectors as inputs, whereas our segment length does vary. To solve this, we summarize each segment in a set of key features, chosen to be as informative as possible to CERN experts when answering `COBRAS`' queries. Furthermore, the evaluation labels, $\{TP, TN, FP, FN\}$, are used in answering COBRAS' questions, but not as input data.

Expert feedback `COBRAS`' questions are answered either automatically, or by a human expert. Suppose the question *"Do segments A and B belong together?"* To answer, we first look at the evaluation labels of segments A and B. One of two possible scenarios unfolds. Scenario one occurs if neither A or B has a FP label. In this case, the question is answered automatically: A and B belong together only if their labels are identical. Scenario two occurs if at least one of the segments was labelled as FP. In that case, the expert takes a closer look and makes the final decision.

Output data Eventually, `COBRAS` outputs a clustering of segments, consistent with expert feedback. This can be thought of as a slight correction of the original output of the anomaly detector. Of course, this only concerns historical data, but this knowledge transfers easily to unseen data. Consider a new segment, with its associated anomaly score. First, using a simple k-nearest neighbour classifier, we pinpoint which cluster it resembles most. Next, we look at the dominant label, L, in that cluster. If $L \in \{TP, FN\}$, the segment is considered anomalous. Alternatively, if $L \in \{TN, FP\}$, we classify the segment as normal.

Results After incorporation of expert feedback for 2017, we observe increased performance of the `GMM` anomaly detector on data of 2018 (Table 5). Additionally, we observe the same effect when using the `iForest` anomaly detector, which originally (i.e. without additional expert knowledge) did not perform well. These encouraging results suggest that adding expert knowledge by means of semi-supervised clustering is a viable approach.

Table 5: Incorporation of expert feedback from 2017 yields improved performance for 2018 with `GMM`.

	anomaly	normal
detected	6	2
undetected	1	1437

CONCLUSION

In this paper, we proposed a proof-of-concept, scalable anomaly detection application for LHC beam transfer installations. This application is structured as a pipeline which includes preprocessing, anomaly detection with ML models, postprocessing, evaluation and an interactive visualization step. Our experiments yielded promising results, including one on taking expert feedback into account.

REFERENCES

[1] CERN, *The Large Hadron Collider*, `https://home.cern/science/accelerators/large-hadron-collider`

[2] M. J. Barnes, L. Ducimetiere, T. Fowler, V. Senaj, and L. Sermeus, “Injection and extraction magnets: kicker magnets,” Mar. 2011. `arXiv:1103.1583`

[3] W. J. Jurasz, “NXCALS - big data logging system - deployment and monitoring. 4th Developers@CERN Forum,” Oct. 2017. `https://cds.cern.ch/record/2290902`

[4] N. Magnin, E. Carlier, B. Goddard, V. Mertens, and J. A. Uythoven, “Internal Post Operation Check System for Kicker Magnet Current Waveforms Surveillance”, in *Proc. ICALEPCS'13*, San Francisco, CA, USA, Oct. 2013, paper MOPPC029, pp. 131-134.

[5] T. Van Craenendonck, S. Dumančić, E. Van Wolputte, and H. Blockeel, “COBRAS: Interactive Clustering with Pair-wise Queries,” in *Advances in Intelligent Data Analysis XVII*, Springer International Publishing, 2018, pp. 353–366.

[6] T. Van Craenendonck, W. Meert, S. Dumančić, and H. Blockeel, “COBRAS TS: A New Approach to Semi-supervised Clustering of Time Series,” in *International Conference on Discovery Science*, Springer, 2018, pp. 179–193.