Anomaly Detection using Machine Learning for Data Quality Monitoring in the CMS Experiment

Author: Agrima Seth
Supervisors: Gianluca Cerminara (CERN)
Adrian Alan Pol (Universite de Paris-Sud 11)

August 18, 2017

Abstract

Reliable, robust and fast-turnaround monitoring of the quality of the data is a key asset to deliver high-quality data for physics analysis for any modern High Energy Physics experiment. The current paradigm of the quality assessment in the CMS collaboration is based on the scrutiny of a large number of histograms by detector experts comparing them with a reference. The project aims at applying recent progress in Machine Learning techniques to the automation of this process allowing the check of large volumes of data in real-time and improving the ability to detect unexpected features. A test implementation using an unsupervised machine learning model focused on the data of one of the CMS muon detectors has been developed and bench-marked on real and fake data.

1 Introduction

The central feature of the CMS apparatus is a superconducting solenoid of 6m internal diameter, providing a magnetic field of 3.8T, within the solenoid volume are a silicon pixel and strip tracker, a lead tungstate crystal electromagnetic calorimeter, and a brass and scintillator hadron calorimeter, each composed of a barrel and two endcap sections. Forward calorimeters extend the pseudorapidity coverage provided by the barrel and endcap detectors.

Muons are measured in the pseudorapidity range $|\eta| < 2.4$, with detection planes made using three technologies: drift tubes (DT), cathode strip chambers, and resistive plate chambers [1]. A more detailed description of the CMS detector, together with a definition of the coordinate system used and the relevant kinematic variables, can be found in [2].

In our project we are concentrating on analysing the occupancy of the drift tube chambers, i.e. number of counts of electronic signal per read out channel, developing an algorithm which could identify chambers manifesting anomalous behaviours. For this we delved into unsupervised machine learning models which can learn hidden correlations in data and spot the anomalous chambers.

1.1 Data Collection

The data used in input to the algorithm are the distribution of the number of hits per channel in each DT chamber as populated by the current Data Quality Monitoring (DQM) infrastructure and served by a web application An example can be seen in Fig 1.

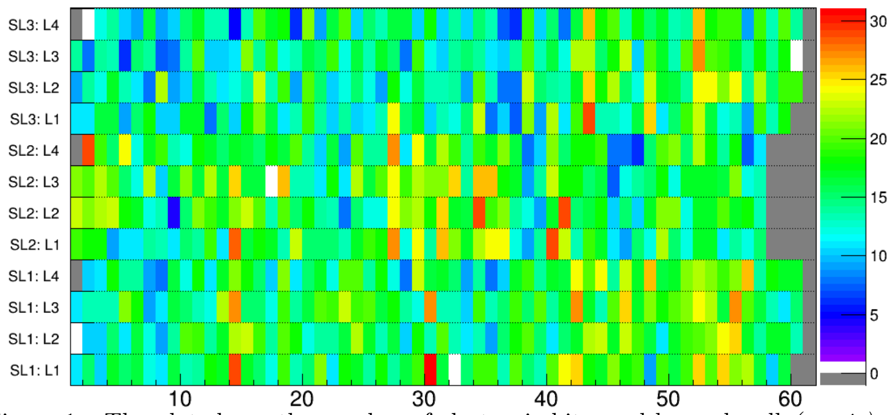

Figure 1: The plot shows the number of electronic hits read by each cell *(x axis)* of the 12 layers *(y axis)* composing a DT chamber.

Only data classified as "usable for physics" and acquired during the 2016 LHC run has been used for training and testing. The dataset collected was composed of 10k datapoints. These were then filtered to remove datapoints corresponding to the outermost chambers (station 4) being the most difficult to model due to radiation background. Further, this dataset was split into two subsets A and B with A containing chambers with no or only one faulty layer and B containing also chambers with more than 1 faulty layer and known anomalies (more details are discussed in the further sections).

2 Data Pre-processing

2.1 Data Segregation

Once sufficient data were collected; it was transformed into a matrix where each chamber was an observation with 15 features to describe it (i.e. n x 15 matrix). The data points were further segregated into three subsets:

- Dataset A (size = 5990, training: 80%, testing: 20 %): it consists of only good chambers from the the collected data; they were labelled using algorithm in [3].
- Dataset B (size = 4000, testing: 100%): it consists of a mix of good chambers and known anomalous chambers. This majorly consisted of data which was rejected by algorithm in [3].
- Dataset C (size = 10, testing: 100%): this was artificially created for further evaluation discussed in Section 3.5.

2.2 Data Normalizations

The data corresponding to each chamber were annotated with the corresponding run number assigned by the CMS data acquisition system. Each run has a different length corresponding to different integration times for the input distributions.

Ensuring standardised feature values implicitly weights all features equally in their representation; hence it is essential to normalize data. Therefore, the data was grouped by run number and then the occupancy values were normalized. The following normalization techniques were used for comparison:

- Normalization by Integration Time: in this approach data from each run was normalized by the integration interval of the occupancy data of the run.

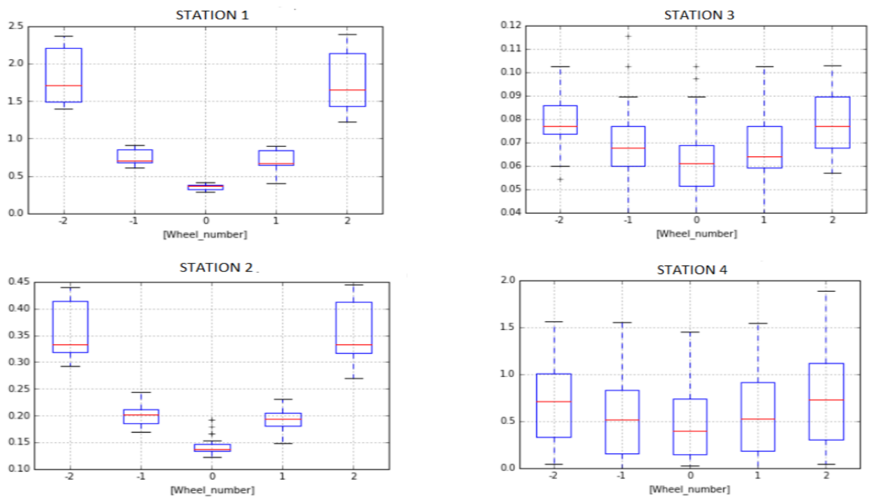

Figure 2: Distribution of time normalized occupancy per wheel for each station for three runs.

  Since this approach did not show significant reduction of variance in the data we opted for other normalization techniques.

- Min-Max Normalization: in this approach, the data is scaled to a fixed range - 0 to 1.
  $X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$ where X is the occupancy of the layer under consideration, X_{min} and X_{max} are the maximum and minimum occupancy for the given run respectively.

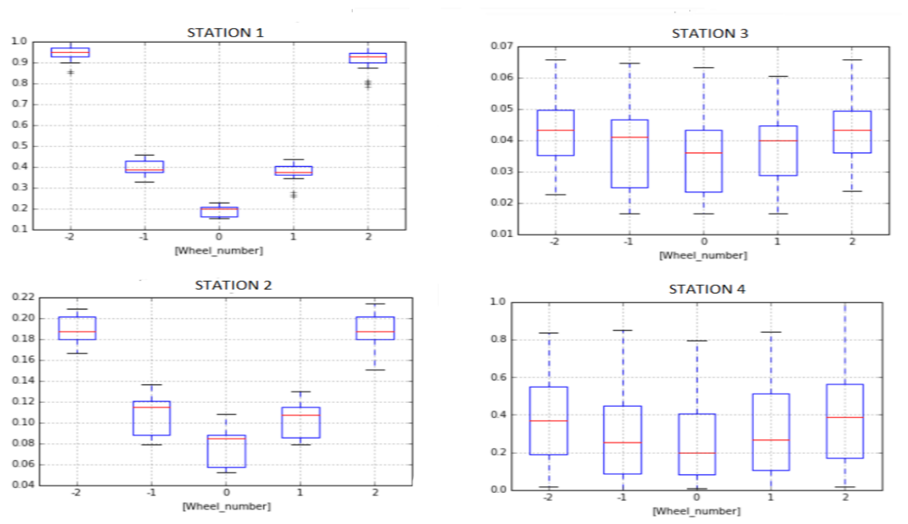

Figure 3: Distribution of min-max normalized occupancy per wheel for each station for three runs.

- Robust Normalization: this scaler removes the median and scales the data according to the quantile range, ensuring that outliers do not influence the sample mean/variance in a negative way. So we decided to test its performance on our data.
  $X_{norm} = \frac{X-Q1}{Q4-Q1}$
  where Q1 and Q4 are the minimum and maximum quartile for the given run respectively.

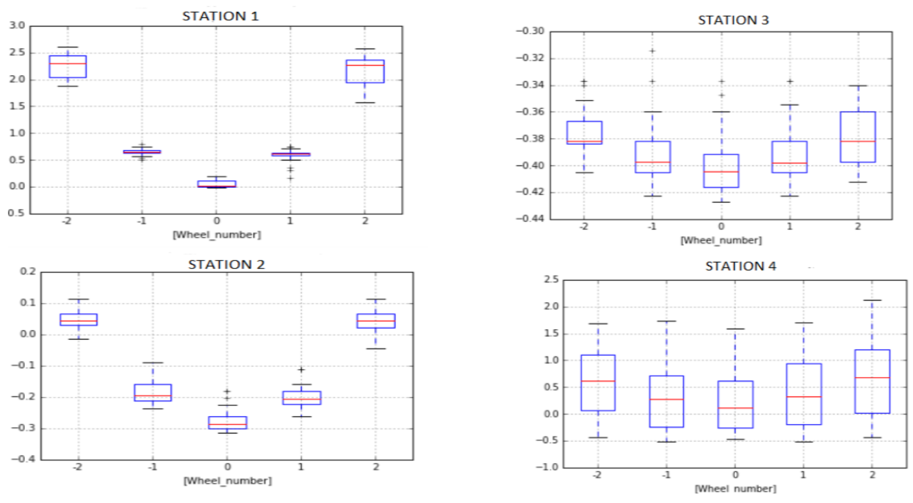

Figure 4: Distribution of robust normalized occupancy per wheel for each station for three runs.

For the first autoencoder architecture we decided to proceed with min-max normalization strategy.

3 Autoencoder Model

An autoencoder is an artificial neural network used for unsupervised learning of efficient codings. The aim of an autoencoder is to learn a representation (encoding) for a set of data [4]. The output layer then tries to reconstruct the input from the learnt representation. Based on the mean square of the reconstructed output and the input; a decision is taken on classifying the chamber as anomalous.

3.1 Terms used

- Topology: position of chamber is defined by wheel number, section number and station number.
- Layer: each chamber is made of 12 layers.

3.2 Feature Selection

There was a comparison between two sets of features:

- Topology and Mean occupancy per layer;
- Topology and Median occupancy per layer.

3.3 Activation Function

In an autoencoder's encoder and decoder part the programmer can use different activation function. We have used *Relu* activation in both the parts.
$h = \max(0, a)$ where $a = Wx + b$ because of its efficient gradient propagation i.e. no vanishing or exploding gradient problems, scale invariance and sparse representations.

3.4 Architecture

For our project we created the following autoencoder architecture:

- Input layer with encoder.
- Single hidden layer with dimensionality 5 and 3 were benchmarked for performance based on the results dimensionality of 5 was chosen for the first iteration of the model; since we were working with a 15 dimensional feature space we did not opt for multiple hidden layers as they would have increased model complexity.
- Output layer with decoder.

3.5 Autoencoder training, testing and evaluation

- For training the autoencoder Dataset A (see Section 2.1) was used. It includes 5990 datapoints and it was split in 80-20 proportion: 80% was used to train the autoencoder and validated with the remaining 20%.

- Once the autoencoder architecture was chosen we tested it with dataset B (see Section 2.1). It consisted of 4000 data points which had a mix of good and known anomalous data-points.
- We created a dataset C containing synthetic anomalous data with anomalies which we were aiming to be captured by our model and would be difficult to spot by the human expert. The autoencoder was evaluated using this dataset.

4 Results

The plots below show the distribution of mean squared error between input and reconstructed values for the two feature sets in section 3.2. Topology and Median occupancy per layer are a better distinguishing feature than Topology and Mean occupancy per layer. It can be inferred from Fig 5 that there is a visible separation between good chambers and the anomalous chambers.

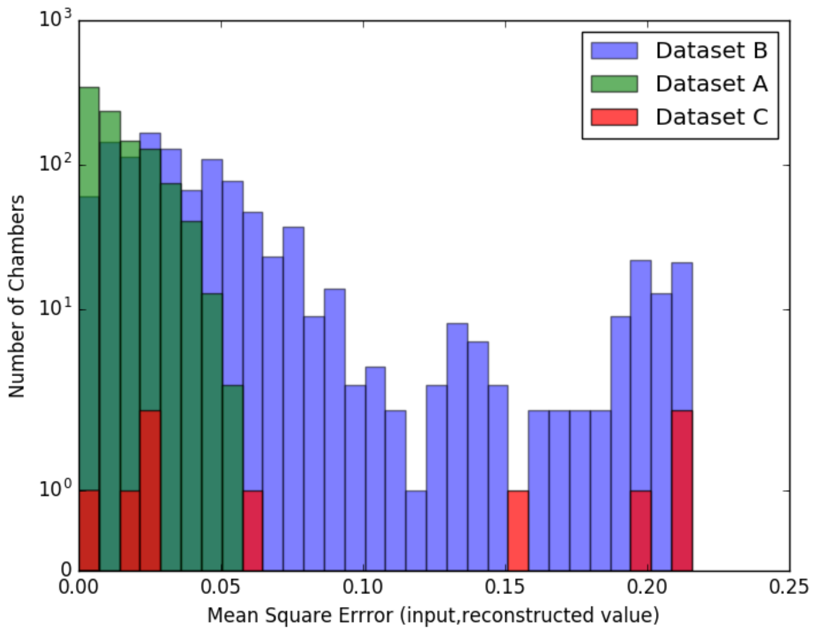

Figure 5: Distribution of Mean Squared Error between input and reconstructed values with topology and Median per layer as features.

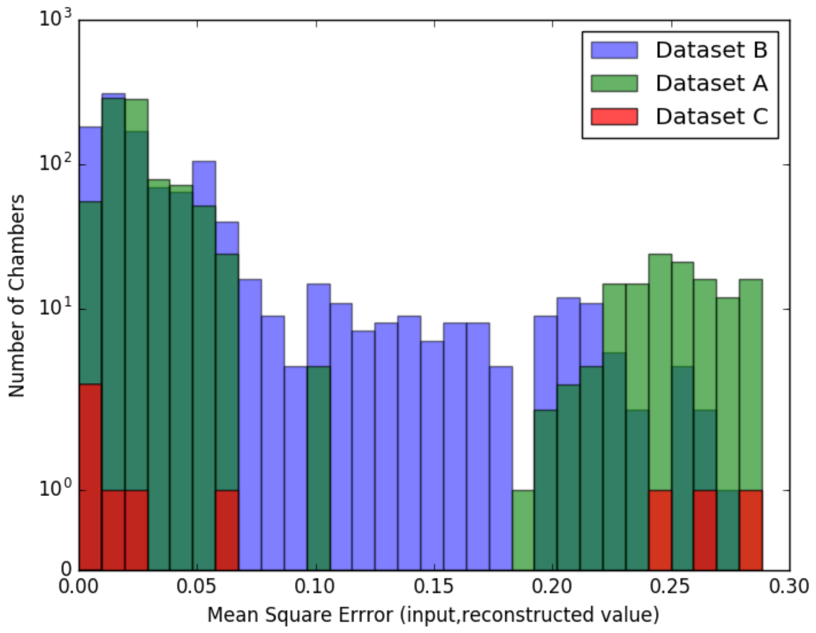

Figure 6: Distribution of Mean Squared Error between input and reconstructed values with topology and Mean per layer as features.

Example of an anomalous and non-anomalous chamber spotted by the autoencoder is show below:

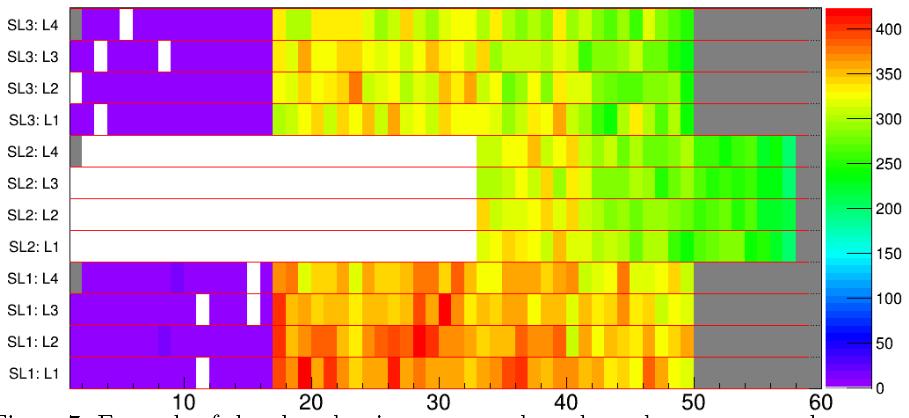

Figure 7: Example of chamber showing an anomalous channel occupancy and successfully identified by the autoencoder.

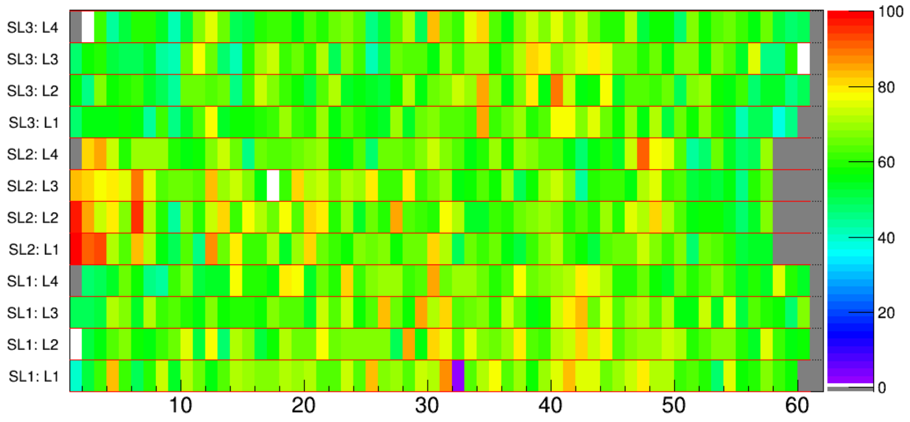

Figure 8: Example of chamber showing a non-anomalous channel occupancy and successfully identified by the autoencoder.

5 Outlook

The first version of the autoencoder, prior to any optimization of its architecture demonstrated to be a promising technique for anomaly detection in Data Quality Monitoring applications. Further development would require the optimization of the autoencoder architecture and enrichment of the feature set. Also, the correct normalization of the data demonstrated to be pivotal for the performance of the algorithm and other strategies, including information on the data acquisition conditions could be exploited.

References

[1] S. Chatrchyan *et al.* [CMS Collaboration], “Performance of CMS muon reconstruction in pp collision events at $\sqrt{s} = 7$ TeV,” JINST **7** (2012) P10002 doi:10.1088/1748-0221/7/10/P10002 [arXiv:1206.4071 [physics.ins-det]].

[2] S. Chatrchyan *et al.* [CMS Collaboration], “The CMS experiment at the CERN LHC,” JINST **3** (2008) S08004 doi:10.1088/1748-0221/3/08/S08004.

[3] `https://github.com/AdrianAlan/DT-Digi-Occupancy/tree/master/notebooks/drift_tubes_digi_occupancy.ipynb`

[4] Bengio, Yoshua, Ian J. Goodfellow, and Aaron Courville. ”Deep learning.” An MIT Press book. (2015) (Pages 500 -502).