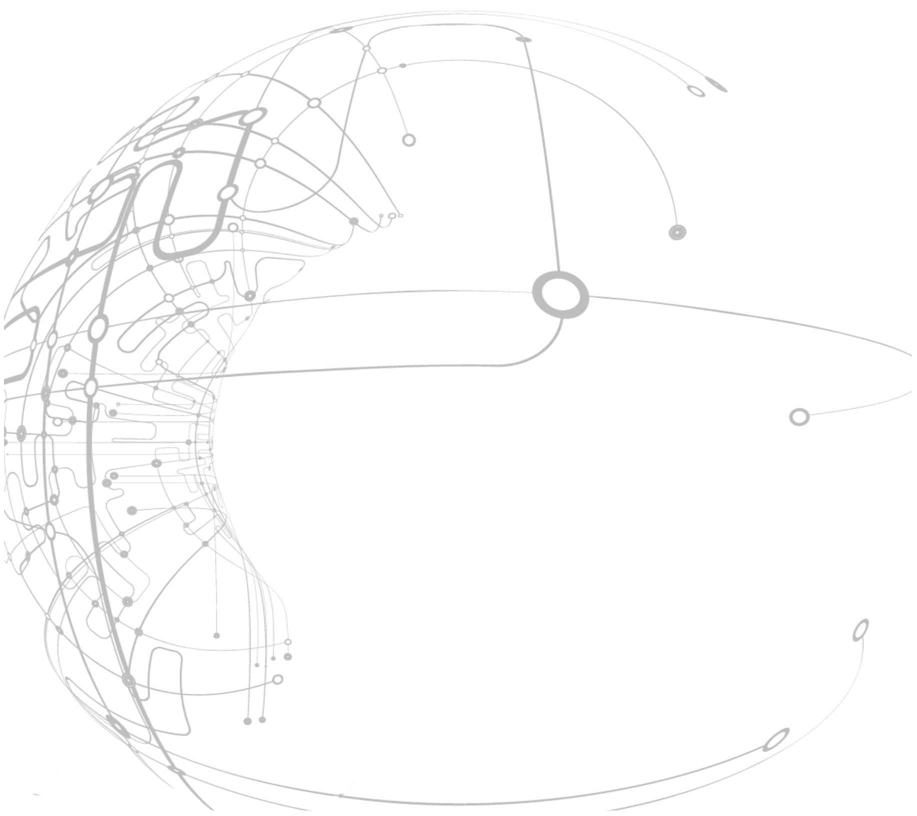


# Data Quality Monitoring at CMS with Machine Learning

***July-August 2016***

*Author:*
*Aytaj Aghabayli*

*Supervisors:*
*Jean-Roch Vlimant*
*Maurizio Pierini*

***CERN openlab Summer Student Report 2016***

## Abstract

The Data Quality Monitoring (DQM) of CMS is a key asset to deliver high-quality data for physics analysis and it is used both in the online and offline environment. The current paradigm of the quality assessment 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 the DQM scrutiny. We explored the landscape of existing ML algorithms with particular attention to supervised problems (for offline DQM) to demonstrate  their validity and usefulness on real test cases using CMS data.

## Table of Contents

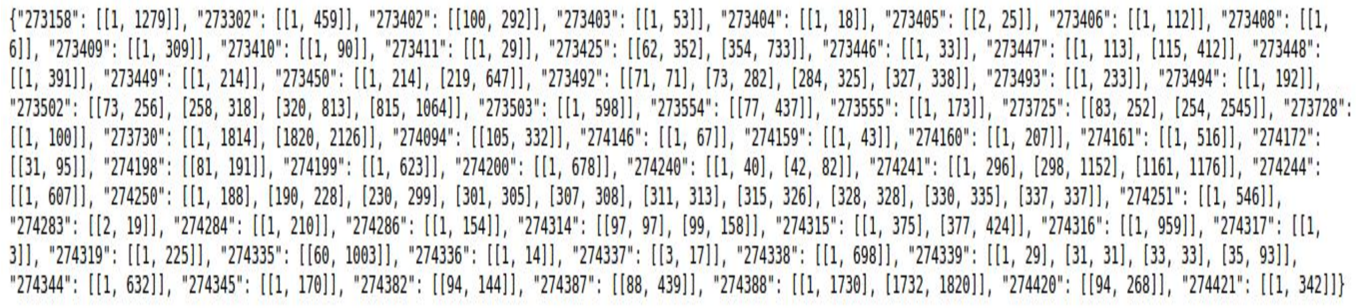
# 1 Introduction

The Data Quality Monitoring (DQM) of the Compact Muon Solenoid (CMS) silicon tracking detectors (Tracker) at the Large Hadron Collider (LHC) at CERN is a software based system designed to monitor the detector and reconstruction performance, to identify problems and to certify the collected data for physics analysis. Its flexibility allowed its integration in two environments: online, for real-time detector monitoring; offline, for the final, fine-grained data certification. Online, DQM is designed to detect real-time response immediately, to analyze any problem and solve it quickly in order to registrate as much good-quality data as possible. Unlike online DQM, offline DQM can additionally rely on high-level quantities, derived during the CPU-expensive event reconstruction (performed in the CMS T0, hosted in the CERN computing center).

To turn the DQM into an automatic system, we reformulated the problem in terms of defining the optimal binary classification algorithm to separate good and bad portions of the data. This is done considering a dataset in which each entry is a luminosity section, the smallest quantum of CMS data, consisting of about 23 seconds of recorded collisions. To train the algorithm, we separate the available data according to the decision taken by experts (the human-based DQM currently in use by CMS). We then train an algorithm to replicate this decision.

# 2 Golden JSON at CMS

The CMS offline data certification process results in a file with the list of good luminosity sections, written in JSON format and usually referred to as "the golden json". An example of golden json file is shown in Fig. 1.



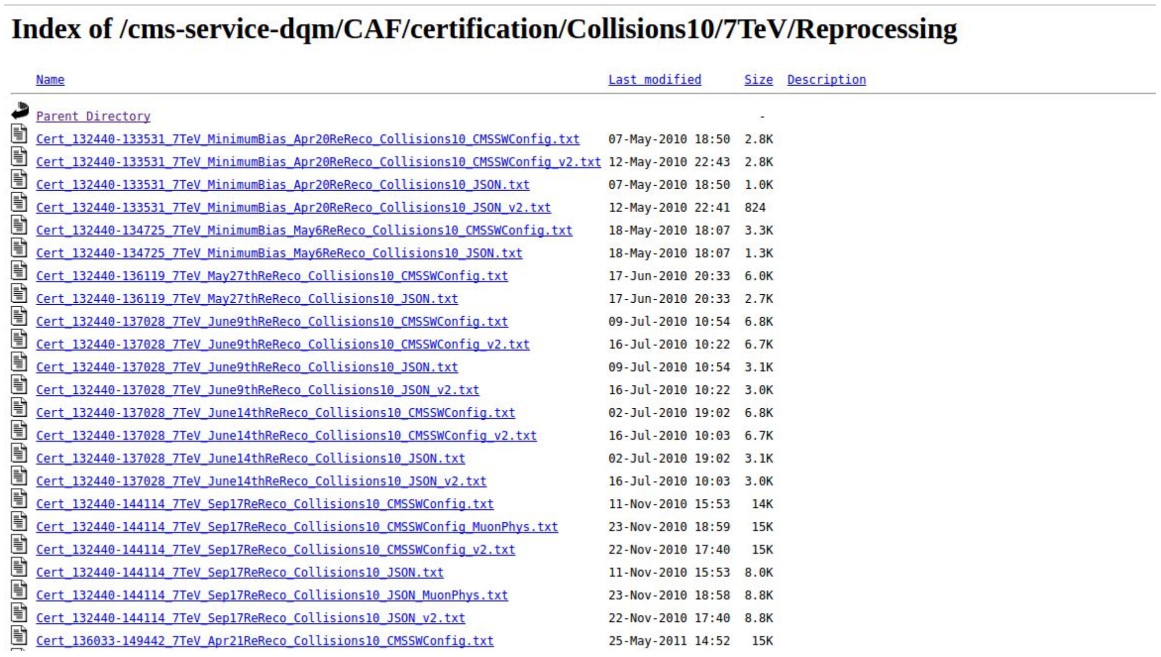
```
{"273158": [[1, 1279]], "273302": [[1, 459]], "273402": [[100, 292]], "273403": [[1, 53]], "273404": [[1, 18]], "273405": [[2, 25]], "273406": [[1, 112]], "273408": [[1,
6]], "273409": [[1, 309]], "273410": [[1, 90]], "273411": [[1, 29]], "273425": [[62, 352], [354, 733]], "273446": [[1, 33]], "273447": [[1, 113], [115, 412]], "273448":
[[1, 391]], "273449": [[1, 214]], "273450": [[1, 214], [219, 647]], "273492": [[71, 71], [73, 282], [284, 325], [327, 338]], "273493": [[1, 233]], "273494": [[1, 192]],
"273502": [[73, 256], [258, 318], [320, 813], [815, 1064]], "273503": [[1, 598]], "273554": [[77, 437]], "273555": [[1, 173]], "273725": [[83, 252], [254, 2545]], "273728":
[[1, 100]], "273730": [[1, 1814], [1820, 2126]], "274094": [[105, 332]], "274146": [[1, 67]], "274159": [[1, 43]], "274160": [[1, 207]], "274161": [[1, 516]], "274172":
[[31, 95]], "274198": [[81, 191]], "274199": [[1, 623]], "274200": [[1, 678]], "274240": [[1, 40], [42, 82]], "274241": [[1, 296], [298, 1152], [1161, 1176]], "274244":
[[1, 607]], "274250": [[1, 188], [190, 228], [230, 299], [301, 305], [307, 308], [311, 313], [315, 326], [328, 328], [330, 335], [337, 337]], "274251": [[1, 546]],
"274283": [[2, 19]], "274284": [[1, 210]], "274286": [[1, 154]], "274314": [[97, 97], [99, 158]], "274315": [[1, 375], [377, 424]], "274316": [[1, 959]], "274317": [[1,
3]], "274319": [[1, 225]], "274335": [[60, 1003]], "274336": [[1, 14]], "274337": [[3, 17]], "274338": [[1, 698]], "274339": [[1, 29], [31, 31], [33, 33], [35, 93]],
"274344": [[1, 632]], "274345": [[1, 170]], "274382": [[94, 144]], "274387": [[88, 439]], "274388": [[1, 1730], [1732, 1820]], "274420": [[94, 268]], "274421": [[1, 342]]}
```

*Fig. 1: example of a golden json file.*

The general format of a CMS json file is:

{"Run Number":[Lumi range, Lumi range, Lumi range, ...],
"Run Number":[Lumi range, Lumi range, Lumi range, ...],
...}

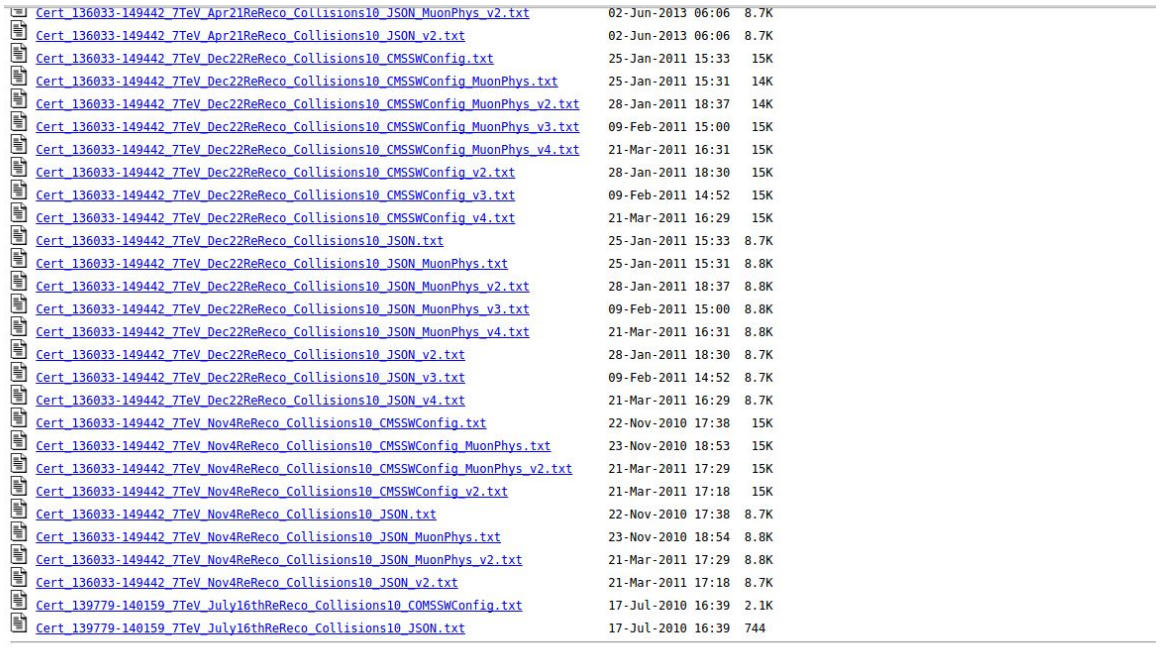
The data collected by CMS are separated into different datasets, based on the output of the online trigger filter.



## Index of /cms-service-dqm/CAF/certification/Collisions10/7TeV/Reprocessing

```
Name                                                                        Last modified      Size  Description

Parent Directory                                                                                  -
Cert_132440-133531_7TeV_MinimumBias_Apr20ReReco_Collisions10_CMSSWConfig.txt      07-May-2010 18:50  2.8K
Cert_132440-133531_7TeV_MinimumBias_Apr20ReReco_Collisions10_CMSSWConfig_v2.txt   12-May-2010 22:43  2.8K
Cert_132440-133531_7TeV_MinimumBias_Apr20ReReco_Collisions10_JSON.txt             07-May-2010 18:50  1.0K
Cert_132440-133531_7TeV_MinimumBias_Apr20ReReco_Collisions10_JSON_v2.txt          12-May-2010 22:41  824
Cert_132440-134725_7TeV_MinimumBias_May6ReReco_Collisions10_CMSSWConfig.txt       18-May-2010 18:07  3.3K
Cert_132440-134725_7TeV_MinimumBias_May6ReReco_Collisions10_JSON.txt              18-May-2010 18:07  1.3K
Cert_132440-136119_7TeV_May27thReReco_Collisions10_CMSSWConfig.txt                17-Jun-2010 20:33  6.0K
Cert_132440-136119_7TeV_May27thReReco_Collisions10_JSON.txt                       17-Jun-2010 20:33  2.7K
Cert_132440-137028_7TeV_June9thReReco_Collisions10_CMSSWConfig.txt                09-Jul-2010 10:54  6.8K
Cert_132440-137028_7TeV_June9thReReco_Collisions10_CMSSWConfig_v2.txt             16-Jul-2010 10:22  6.7K
Cert_132440-137028_7TeV_June9thReReco_Collisions10_JSON.txt                       09-Jul-2010 10:54  3.1K
Cert_132440-137028_7TeV_June9thReReco_Collisions10_JSON_v2.txt                    16-Jul-2010 10:22  3.0K
Cert_132440-137028_7TeV_June14thReReco_Collisions10_CMSSWConfig.txt               02-Jul-2010 19:02  6.8K
Cert_132440-137028_7TeV_June14thReReco_Collisions10_CMSSWConfig_v2.txt            16-Jul-2010 10:03  6.7K
Cert_132440-137028_7TeV_June14thReReco_Collisions10_JSON.txt                      02-Jul-2010 19:02  3.1K
Cert_132440-137028_7TeV_June14thReReco_Collisions10_JSON_v2.txt                   16-Jul-2010 10:03  3.0K
Cert_132440-144114_7TeV_Sep17ReReco_Collisions10_CMSSWConfig.txt                  11-Nov-2010 15:53   14K
Cert_132440-144114_7TeV_Sep17ReReco_Collisions10_CMSSWConfig_MuonPhys.txt         23-Nov-2010 18:59   15K
Cert_132440-144114_7TeV_Sep17ReReco_Collisions10_CMSSWConfig_v2.txt               22-Nov-2010 17:40   15K
Cert_132440-144114_7TeV_Sep17ReReco_Collisions10_JSON.txt                         11-Nov-2010 15:53  8.0K
Cert_132440-144114_7TeV_Sep17ReReco_Collisions10_JSON_MuonPhys.txt                23-Nov-2010 18:58  8.8K
Cert_132440-144114_7TeV_Sep17ReReco_Collisions10_JSON_v2.txt                      22-Nov-2010 17:40  8.8K
Cert_136033-149442_7TeV_Apr21ReReco_Collisions10_CMSSWConfig.txt                  25-May-2011 14:52   15K
```



```
Cert_136033-149442_7TeV_Apr21ReReco_Collisions10_JSON_MuonPhys_v2.txt             02-Jun-2013 06:06  8.7K
Cert_136033-149442_7TeV_Apr21ReReco_Collisions10_JSON_v2.txt                      02-Jun-2013 06:06  8.7K
Cert_136033-149442_7TeV_Dec22ReReco_Collisions10_CMSSWConfig.txt                  25-Jan-2011 15:33   15K
Cert_136033-149442_7TeV_Dec22ReReco_Collisions10_CMSSWConfig_MuonPhys.txt         25-Jan-2011 15:31   14K
Cert_136033-149442_7TeV_Dec22ReReco_Collisions10_CMSSWConfig_MuonPhys_v2.txt      28-Jan-2011 18:37   14K
Cert_136033-149442_7TeV_Dec22ReReco_Collisions10_CMSSWConfig_MuonPhys_v3.txt      09-Feb-2011 15:00   15K
Cert_136033-149442_7TeV_Dec22ReReco_Collisions10_CMSSWConfig_MuonPhys_v4.txt      21-Mar-2011 16:31   15K
Cert_136033-149442_7TeV_Dec22ReReco_Collisions10_CMSSWConfig_v2.txt               28-Jan-2011 18:30   15K
Cert_136033-149442_7TeV_Dec22ReReco_Collisions10_CMSSWConfig_v3.txt               09-Feb-2011 14:52   15K
Cert_136033-149442_7TeV_Dec22ReReco_Collisions10_CMSSWConfig_v4.txt               21-Mar-2011 16:29   15K
Cert_136033-149442_7TeV_Dec22ReReco_Collisions10_JSON.txt                         25-Jan-2011 15:33  8.7K
Cert_136033-149442_7TeV_Dec22ReReco_Collisions10_JSON_MuonPhys.txt                25-Jan-2011 15:31  8.8K
Cert_136033-149442_7TeV_Dec22ReReco_Collisions10_JSON_MuonPhys_v2.txt             28-Jan-2011 18:37  8.8K
Cert_136033-149442_7TeV_Dec22ReReco_Collisions10_JSON_MuonPhys_v3.txt             09-Feb-2011 15:00  8.8K
Cert_136033-149442_7TeV_Dec22ReReco_Collisions10_JSON_MuonPhys_v4.txt             21-Mar-2011 16:31  8.8K
Cert_136033-149442_7TeV_Dec22ReReco_Collisions10_JSON_v2.txt                      28-Jan-2011 18:30  8.7K
Cert_136033-149442_7TeV_Dec22ReReco_Collisions10_JSON_v3.txt                      09-Feb-2011 14:52  8.7K
Cert_136033-149442_7TeV_Dec22ReReco_Collisions10_JSON_v4.txt                      21-Mar-2011 16:29  8.7K
Cert_136033-149442_7TeV_Nov4ReReco_Collisions10_CMSSWConfig.txt                   22-Nov-2010 17:38   15K
Cert_136033-149442_7TeV_Nov4ReReco_Collisions10_CMSSWConfig_MuonPhys.txt          23-Nov-2010 18:53   15K
Cert_136033-149442_7TeV_Nov4ReReco_Collisions10_CMSSWConfig_MuonPhys_v2.txt       21-Mar-2011 17:29   15K
Cert_136033-149442_7TeV_Nov4ReReco_Collisions10_CMSSWConfig_v2.txt                21-Mar-2011 17:18   15K
Cert_136033-149442_7TeV_Nov4ReReco_Collisions10_JSON.txt                          22-Nov-2010 17:38  8.7K
Cert_136033-149442_7TeV_Nov4ReReco_Collisions10_JSON_MuonPhys.txt                 23-Nov-2010 18:54  8.8K
Cert_136033-149442_7TeV_Nov4ReReco_Collisions10_JSON_MuonPhys_v2.txt              21-Mar-2011 17:29  8.8K
Cert_136033-149442_7TeV_Nov4ReReco_Collisions10_JSON_v2.txt                       21-Mar-2011 17:18  8.7K
Cert_139779-140159_7TeV_July16thReReco_Collisions10_COMSSWConfig.txt              17-Jul-2010 16:39  2.1K
Cert_139779-140159_7TeV_July16thReReco_Collisions10_JSON.txt                      17-Jul-2010 16:39  744
```

*Fig. 2: List of JSON files*

A data tier may contain multiple data formats, as mentioned below for reconstructed data (See Fig. 3).

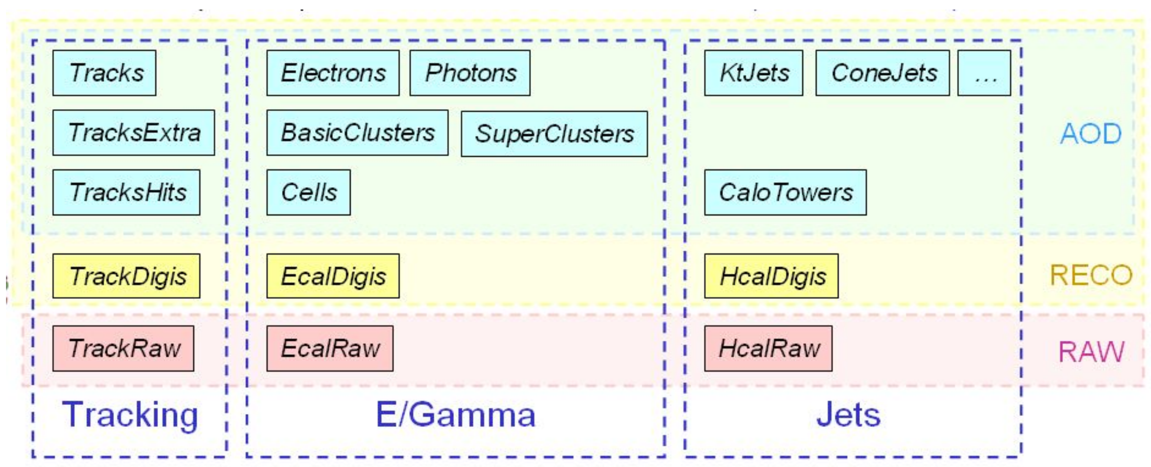


*Fig. 3:* Data tiers

For our exercise, we used the list of luminosity sections officially listed as good. For the bad luminosity sections, we created a dedicated list accessing the CMS run registry and selecting data periods for which any sub detector was excluded for the run. The causes of the exclusion could be of different kind. This is the most general definition of "bad" that we could define through code. Yandex team has experienced on the CMS public data, so that my task was to apply, what they did within CMS.

## 3 DQM system of Yandex team

The classification problem is that each lumi section can belong to one of 3 groups, which are 'white zone', 'black zone' and 'grey zone'. The DQM system, defined by the Yandex team, on which we worked, consist of automatized data quality system for the CMS experiment. This system learns to predict experts' decisions who labeled data collected by CMS as "good" and "bad". The system continuously learns from proficients and leave only non-trivial cases of data for them. A tunable fraction of the data is classified as good and bad, the acceptable mislabeling rate being a tunable parameter at training time. The unlabeled data are then returned to experts for scrutiny. While this formulation does not remove the need of a human operator, it strongly reduce the amount of work required. It is easy to train very accurate classifiers while keeping the fraction of unlabeled data below 50%. The algorithm, used by the Yandex, team is a supervised algorithm consisting of a Random Forest classifier. Three datasets are taken as input: Minimum Bias, Muon, Photon streams. For each dataset, a specific list of features is used. For the main particles in the dataset (e.g., muons for the muon dataset) $p_T$, $\eta$, $\varphi$, m, $f_{x,y,z}$ and additional are total momentum, number of events, etc. From the distribution of this quantities, quantiles at 0, 0.25, 0.5, 0.75, 1 + mean + sigma are extracted.

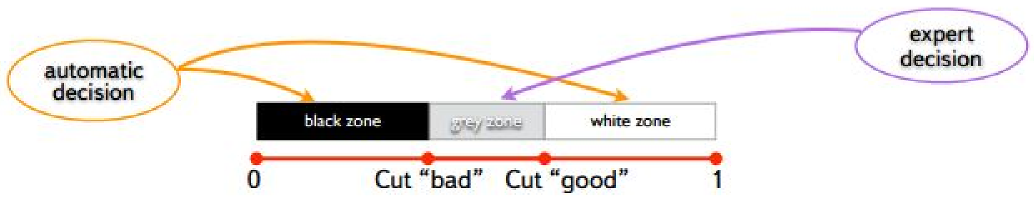


*Fig. 4: DQM system classification principle;*

More information can be found here https://indico.cern.ch/event/532992/contributions/2224631/attachments/1303860/1947683/anomaly-detection-yandex.pdf .

We consider the following quantities:

Rejection Rate = Rejected / Positive + Negative + Rejected;

Pollution Rate = False Positive / True Positive + False Positive = 1 - precision;

Loss Rate = False Negative / True Positive + False Negative = 1 - recall;

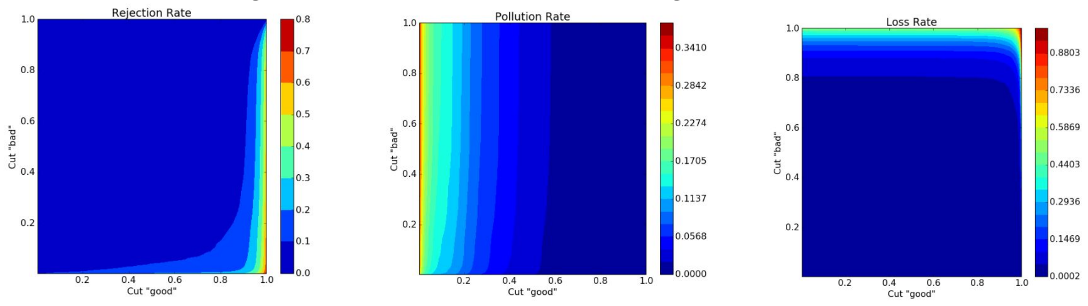


*Fig. 5: Rejection Rate, Polutin Rate, Loss Rate;*

## 4 Applying DQM system to 2010 data of CMS

In this section, we describe the different steps taken to apply the Yandex classification algorithm to the 2010 CMS data: loading data, extraction features, merging data, applying learning algorithm, testing data and analyzing of ROC Curves.

### 4.1 Loading data

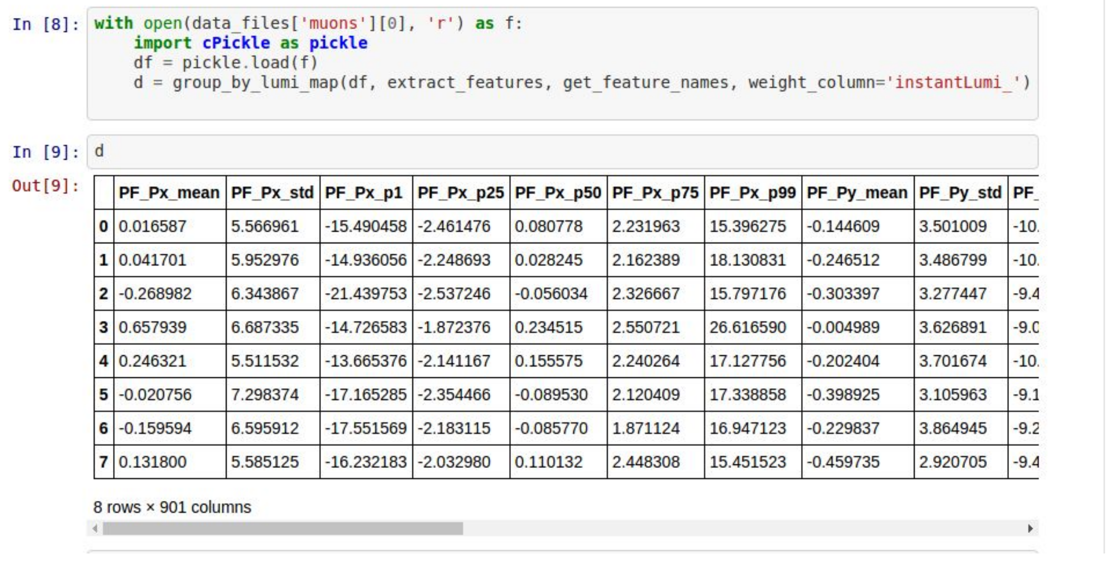
First of all, we imported JSON files contained link of data, which represented root files. Data are loaded from root files, by running the sketch load.py, which have been written in python. All existed lumi sections, also needed features were read, where branch sizes were less than considered in respective config files.

This process contains changing data format from "root" to "pickle", which took about one week and was one of the limiting factors for the effectiveness of this study. Data loading commands have the following structure: "data-extraction/bin/load.py <config> <URL list> <output directory>". Config files are JSON files, which have been divided into 2 groups: per_lumisection and per_event. An example of config files with more details is given below in paragraph 5.

## 4.2 Features extraction

The piece of code, which is shown below, has written on python and contains the process of extraction of features. Percentiles were taken as 1, 25, 50, 75, 99 and extracted given features such as $p_T$, η, φ, m, $f_{x,y,z}$. Out 9 shows all needed features, which we used.

```
percentiles = [1, 25, 50, 75, 99]
def extract_features(block_data, weights = None):
    n_features_per_column = len(percentiles) + 2
    n = block_data.shape[1]
    n_features = n * n_features_per_column + 2
    result = np.ndarray(shape = n_features, dtype='float32')
    for j in xrange(block_data.shape[1]):
        x = block_data[:, j]
        offset = j * n_features_per_column
        result[offset] = np.mean(x)
        result[offset + 1] = np.std(x)
        for i, q in enumerate(percentiles):
            result[offset + i + 2] = np.percentile(x, q = q)
    result[-2] = block_data.shape[0]
    result[-1] = np.mean(weights > 0.0)
    return result
```



```
In [8]: with open(data_files['muons'][0], 'r') as f:
            import cPickle as pickle
            df = pickle.load(f)
            d = group_by_lumi_map(df, extract_features, get_feature_names, weight_column='instantLumi_')

In [9]: d

Out[9]:
```

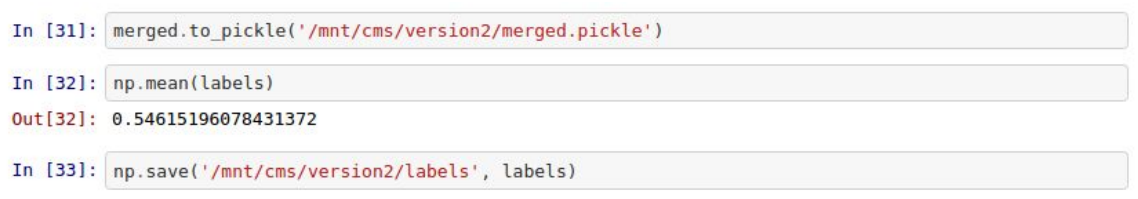
| | PF_Px_mean | PF_Px_std | PF_Px_p1 | PF_Px_p25 | PF_Px_p50 | PF_Px_p75 | PF_Px_p99 | PF_Py_mean | PF_Py_std | PF_ |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.016587 | 5.566961 | -15.490458 | -2.461476 | 0.080778 | 2.231963 | 15.396275 | -0.144609 | 3.501009 | -10. |
| 1 | 0.041701 | 5.952976 | -14.936056 | -2.248693 | 0.028245 | 2.162389 | 18.130831 | -0.246512 | 3.486799 | -10. |
| 2 | -0.268982 | 6.343867 | -21.439753 | -2.537246 | -0.056034 | 2.326667 | 15.797176 | -0.303397 | 3.277447 | -9.4 |
| 3 | 0.657939 | 6.687335 | -14.726583 | -1.872376 | 0.234515 | 2.550721 | 26.616590 | -0.004989 | 3.626891 | -9.0 |
| 4 | 0.246321 | 5.511532 | -13.665376 | -2.141167 | 0.155575 | 2.240264 | 17.127756 | -0.202404 | 3.701674 | -10. |
| 5 | -0.020756 | 7.298374 | -17.165285 | -2.354466 | -0.089530 | 2.120409 | 17.338858 | -0.398925 | 3.105963 | -9.1 |
| 6 | -0.159594 | 6.595912 | -17.551569 | -2.183115 | -0.085770 | 1.871124 | 16.947123 | -0.229837 | 3.864945 | -9.2 |
| 7 | 0.131800 | 5.585125 | -16.232183 | -2.032980 | 0.110132 | 2.448308 | 15.451523 | -0.459735 | 2.920705 | -9.4 |

8 rows × 901 columns

*Fig. 7: Features of data*

## 4.3 Data merging

In the process of merging data, we took all features, joined them together and saved results in files "merged.pickle" as data with pickle format. In addition, extracted labels and saved them in "labels.npy" file as numpy array data. As conclusion, both of them we used in the classification of CMS data, which is described in the paragraph 4.4.



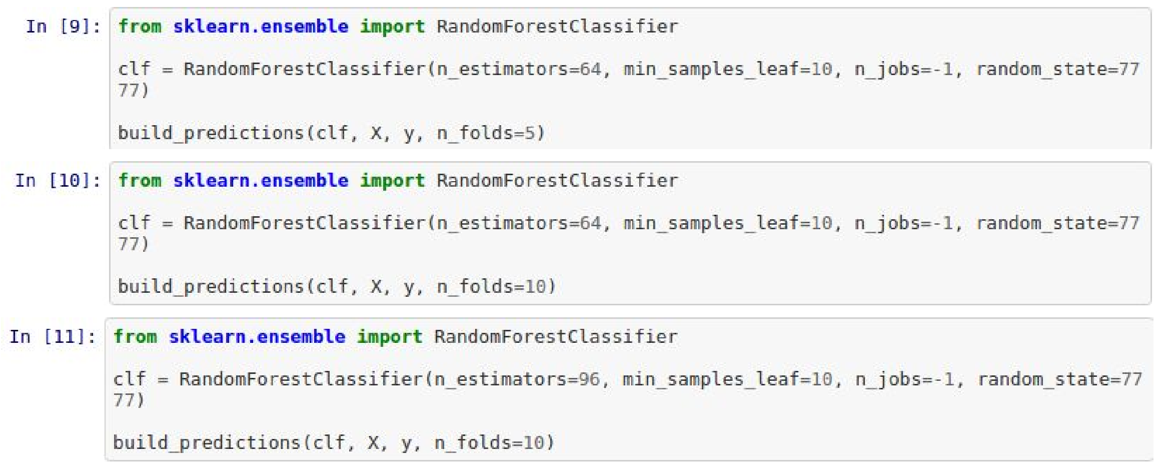
```
In [31]: merged.to_pickle('/mnt/cms/version2/merged.pickle')

In [32]: np.mean(labels)
Out[32]: 0.54615196078431372

In [33]: np.save('/mnt/cms/version2/labels', labels)
```

*Fig. 8: Data merging*

## 4.4 Applying learning algorithm



```
In [9]:  from sklearn.ensemble import RandomForestClassifier

         clf = RandomForestClassifier(n_estimators=64, min_samples_leaf=10, n_jobs=-1, random_state=77
         77)

         build_predictions(clf, X, y, n_folds=5)

In [10]: from sklearn.ensemble import RandomForestClassifier

         clf = RandomForestClassifier(n_estimators=64, min_samples_leaf=10, n_jobs=-1, random_state=77
         77)

         build_predictions(clf, X, y, n_folds=10)

In [11]: from sklearn.ensemble import RandomForestClassifier

         clf = RandomForestClassifier(n_estimators=96, min_samples_leaf=10, n_jobs=-1, random_state=77
         77)

         build_predictions(clf, X, y, n_folds=10)
```

*Fig. 9: Classification function*

As we said before the system is using Random Forest Classifier. In the Fig. 9 there is classifier with different number of estimators and the results are different, what we will see on ROC curves.

## 4.5 ROC curves

On ROC curves we can quantify the accuracy of the trained algorithm. From the first figure is shown below, we can see that the true positive rate tends to 1.0, which means almost all good lumi sections were predicted correctly, also the fact, that the false positive rate does not depart from 0.0, confirmed the same. As conclusion, accuracy of classification is taken as 1.00, which means white and black data sets contain correct data and just little part of them needs decision of experts.

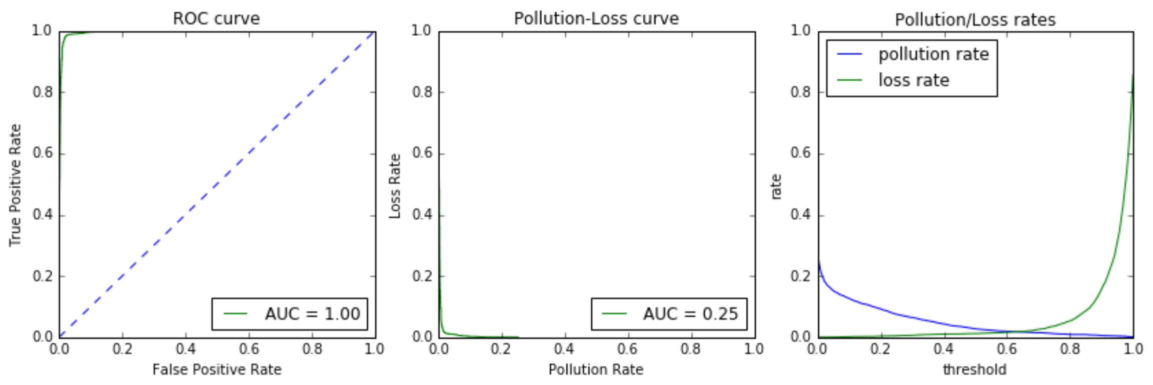


*Fig. 10: ROC curves, Loss rates, Pollution rates*

## 5 Structure of 2016 CMS data and future work

To order to apply this strategy to 2016 data, one needs to update the list of features. This is done editing the config files, taken as input by the Yandex python scripts. These changes reflect the changes in the collections stored by CMS in their data. Below you can see the “branches_minibias.config” file of 2010 data.

```
{
      "per_lumisection" : [
        "EventAuxiliary.id_.run_",
        "EventAuxiliary.id_.luminosityBlock_",
        "EventAuxiliary.time_.timeHigh_",
        "EventAuxiliary.time_.timeLow_",
        "LumiScalerss_scalersRawToDigi__RECO.obj.instantLumi_"
      ],

      "per_event" : {
        "calo" : {
          "read_each" : 2,
          "batch" : 64,
          "branches" : [
            "recoCaloJets_ak5CaloJets__RECO.obj.pt_",
            "recoCaloJets_ak5CaloJets__RECO.obj.eta_",
            "recoCaloJets_ak5CaloJets__RECO.obj.phi_",
            "recoCaloJets_ak5CaloJets__RECO.obj.mass_",
            "recoCaloJets_ak5CaloJets__RECO.obj.vertex_.fCoordinates.fX",
            "recoCaloJets_ak5CaloJets__RECO.obj.vertex_.fCoordinates.fY",
            "recoCaloJets_ak5CaloJets__RECO.obj.vertex_.fCoordinates.fZ"
          ]
        },

        "photons" : {
          "read_each" : 2,
          "batch" : 32,
          "branches" : [
            "recoPhotons_photons__RECO.obj.pt_",
            "recoPhotons_photons__RECO.obj.eta_",
            "recoPhotons_photons__RECO.obj.phi_",
            "recoPhotons_photons__RECO.obj.vertex_.fCoordinates.fX",
            "recoPhotons_photons__RECO.obj.vertex_.fCoordinates.fY",
            "recoPhotons_photons__RECO.obj.vertex_.fCoordinates.fZ"
          ]
        },


        "muons" : {
          "read_each" : 2,
          "batch" : 32,
          "branches" : [
            "recoMuons_muons__RECO.obj.pt_",
            "recoMuons_muons__RECO.obj.eta_",
            "recoMuons_muons__RECO.obj.phi_",
            "recoMuons_muons__RECO.obj.mass_",
            "recoMuons_muons__RECO.obj.vertex_.fCoordinates.fX",
            "recoMuons_muons__RECO.obj.vertex_.fCoordinates.fY",
            "recoMuons_muons__RECO.obj.vertex_.fCoordinates.fZ"
          ]
        },


        "PF" : {
          "read_each" : 8,
          "batch" : 1536,
          "branches" : [
            "recoPFCandidates_particleFlow__RECO.obj.pt_",
            "recoPFCandidates_particleFlow__RECO.obj.eta_",
            "recoPFCandidates_particleFlow__RECO.obj.phi_"
          ]
        }
       }
      }
```

We also read as input the new data, not from the EOS Open Data repository but from the private CMS eos area, where the data were temporarily stored for this exercise to be performed. Attempt to read the data remotely did not succeed. I would like to mention that some information was mixing, notably the instantaneous luminosity of the LHC, which serves to equalize the content of different luminosity section. This raised an issue that the CMS collaboration is now trying to fix.

In the future, once this problem is solved, one could add new features, try other learning algorithms and change existed classification to multiclass classification for each bad JSON, which we already created. Main purpose of this is to analyze how different changes will affect to quality of real CMS data.