Sophisticated anomaly detection

CERN openlab

project for Data Certification with

Yandex School of Economics

M. Borisyak,
D. Derkach,
O. Koval,
F. Ratnikov,
A. Ustyuzhanin

CMS

V. Azzolini,
G. Cerminara,
F. De Guio,
G. Franzoni,
M. Pierini,
A. Pol,
F. Siroky,
J-R. Vlimant

Formalising the Problem

GOAL

to develop machine learning model which will:

◊ improve the accuracy of Data Quality
◊ increase the speed at which this is achieved
◊ limit person power
◊ anticipate important events
◊ adapt behavior in response to changes in data content, user needs or available resources

State of art of the project

◊ Design of a system to assist Data Quality managers by filtering most obvious cases. It automatically classifies marginal cases in general: both of 'good' and 'bad' data, and use human expert decision to classify remaining 'grey area' cases.

◊ Deep learning method for inferring cause of data anomalies. Model learns to identify 'channels' which are affected by an anomaly.

◊ Developing algorithms on 2016 data.

◊ Developing algorithms on 2016 data, first encouraging tests result of anomaly detection and indication of the main feature offender using Auto-Encoders.

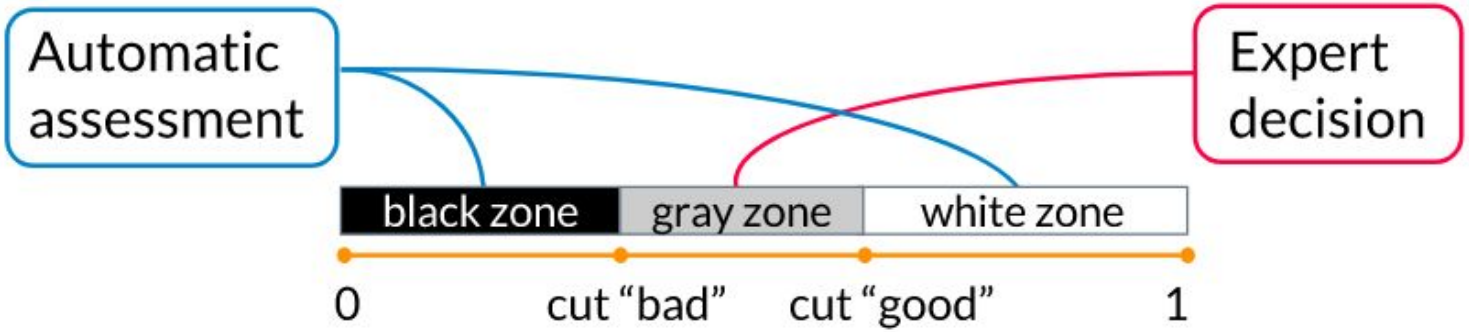
◊ Acting as facilitator for the the whole project. Current data access policies allow technical access to data in real time

Automated data quality system supervised learning approach

◊ Train classifier * on training part of data

Using historical data processed by experts (metadata of “good” or “bad” runs) as input

◊ System learns to classify in 3 categories:



◊ <u>System Goal:</u>

$$\text{Rejection Rate} = \frac{\text{Rejected}}{\text{Total}} \rightarrow \min$$

(Manual Work)

under constraints:

$$\text{Loss Rate} = \frac{\text{False Negative}}{\text{True Positive} + \text{False Negative}} \leq L_0$$

$$\text{Pollution Rate} = \frac{\text{False Positive}}{\text{False Positive} + \text{True Positive}} \leq P_0$$

*Gradient Tree Boosting classifier +
10-fold cross validation scheme to estimate cuts

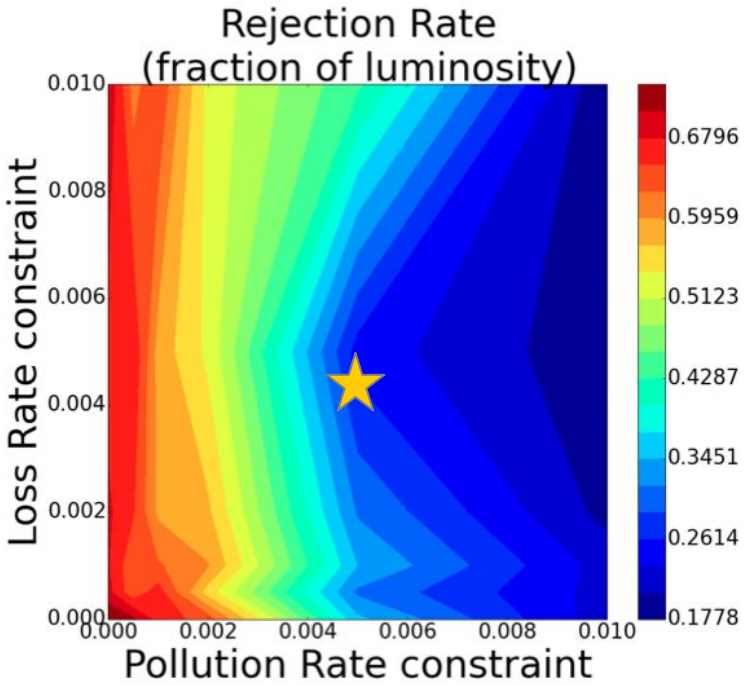
Data & Results

Input:

2010B CMS OpenData

Each lumisection is described by a fixed number of features **~ 2500**, each feature having a hierarchical description:

- ◊ CMS streams: Minimal Bias, Muons, Photons
- ◊ Channels: Muons, Photons, Particle Flow Jets, Calo Jets
- ◊ Quantile by Pt: 0.2, 0.4, 0.6, 0.8, 1.
- ◊ Physics properties: pT, η, φ, Vx, Vy,Vz, mass
- ◊ Statistics within LS for each feature:
  0.1, 0.25, 0.5, 0.75, 0.99 percentiles, mean, variance



Performance:

20% saved person power for Pollution and Loss rates 0.05%

80% saved person power for Pollution and Loss rates 0.5 %

Presented CHEP 2016 San Francisco:
M.Borisyak et al, Towards automation of data quality system for CERN CMS experiment,
https://indico.cern.ch/event/505613/contributions/2227973/

Channel decomposition

Data comes from different sub-detectors or other subsystems, and the global data quality depends on the combinatorial performance of each of them.

- ◊ what channels are responsible for anomalies?
- ◊ if only photons are affected, may muon data still be used?
- ◊ which plots should receive more attention from Data Quality experts?

→ study effect of anomalies on individual channels

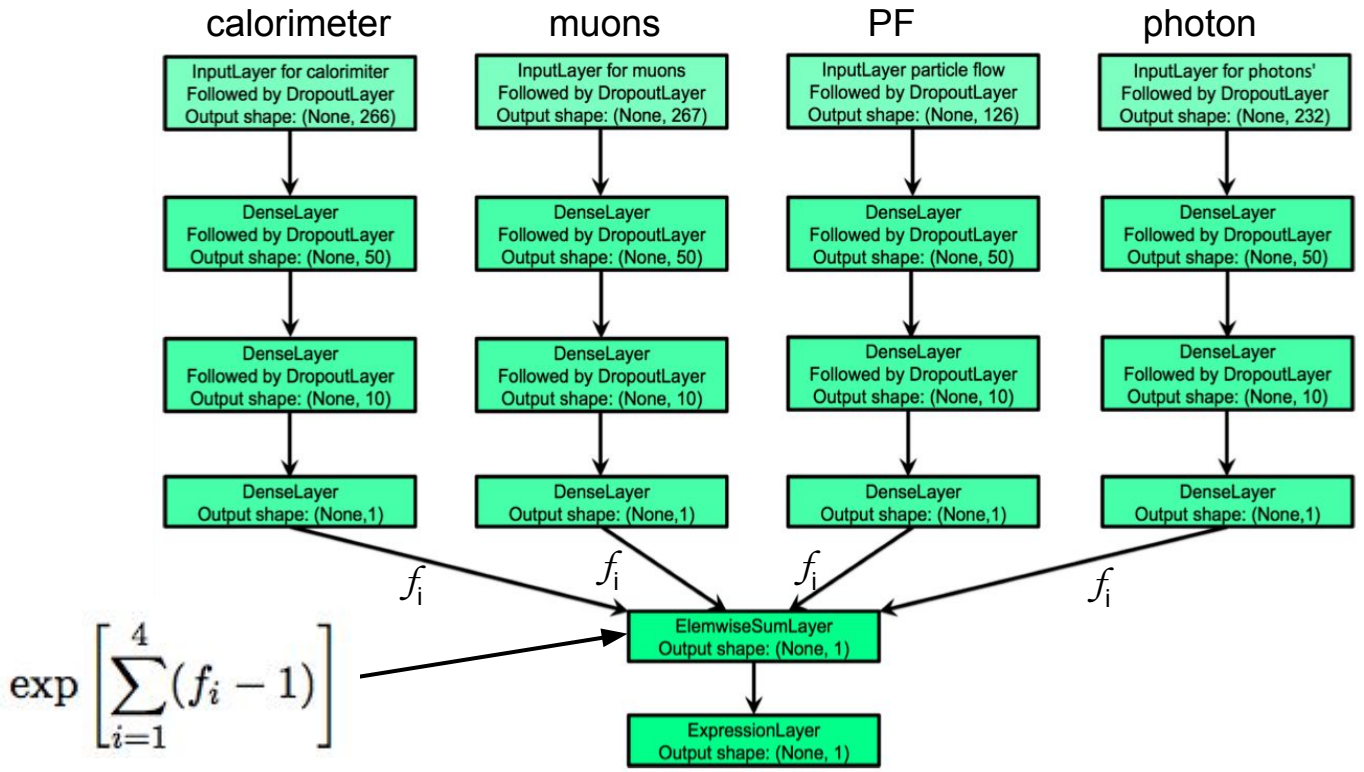
In 2017 work, presented at ACAT conference*, we aim to determine which sub-detector is responsible for anomaly in the detector behaviour.

Knowing only global tag for the whole CMS detector model restores quality of data for each channel separately.

*: https://indico.cern.ch/event/567550/contributions/2629718/

Deep learning method

Special multi-head NN to predict a probability of anomaly in the channels separately



All the sub-networks return scores for their channels.
They are connected via a 'fuzzy AND' operator *
for training

The main novelty of the method is that the model does not require ground truth labels for each channel, only global flag is used for NN training.

* Proof for 'Fuzzy AND' operator decomposition properties

Channel decomposition results

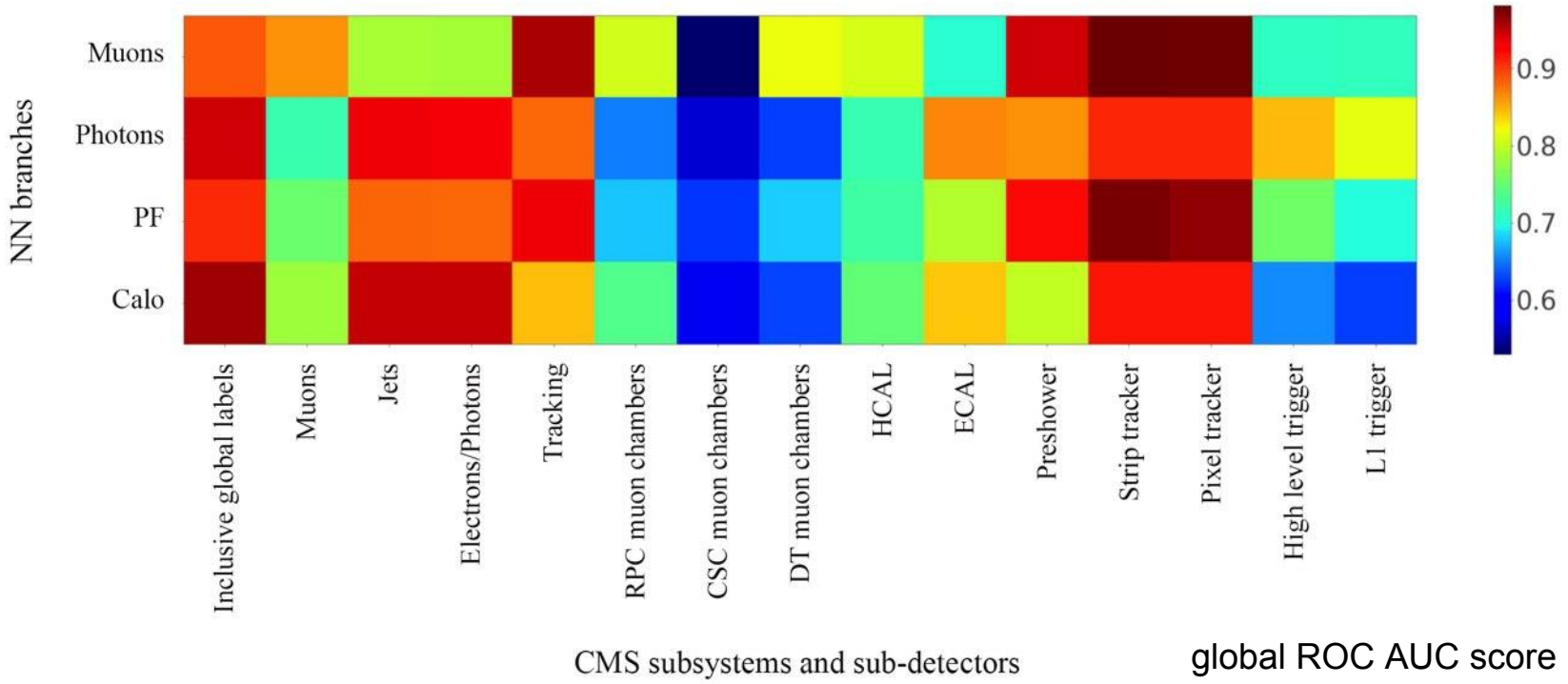
Input: **2010B CMS data** separated between channels inputs

Interpretation:

◊ 'good' samples scores close to 1.

◊ 'bad' samples has two options:
. visible from channel (score ~ 0)
. not visible (score ~ 1)
this data is not affected by an anomaly
→ still useful for physical analysis

Clear correlation between subnetworks' outputs and corresponding subsystem labels



global ROC AUC score = 0.96

2016 CMS data

◊ Overall better quality of 2016 CMS data than 2010

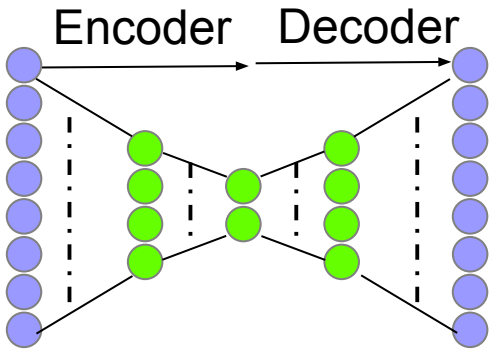
less anomalies to detect, less effective is a supervised technique due to class imbalance. More problems!

◊ Change the lengthy process of collecting, identifying, cleansing and normalizing data, ensuring that a model developed today not only can mimic data quality prescription, but will make good predictions on future data which will be different than the data we're studying to build the model

→ Semi-supervised learning approach
first impressive results with Auto-Encoders *

* this part of the ml4dc project principally CMS internally developed but always with the fruitful collaboration of the Yandex group

Method



$E_X(X - g(f(X)))^2 \rightarrow \min$

Auto-Encoder output is a version or "reconstruction" of the input data.

Unsupervised ML
AE learns its parameter by Minimization of reconstruction error between input and output

As good instances occur much more frequently than anomalous instances.
Train model only on good data.

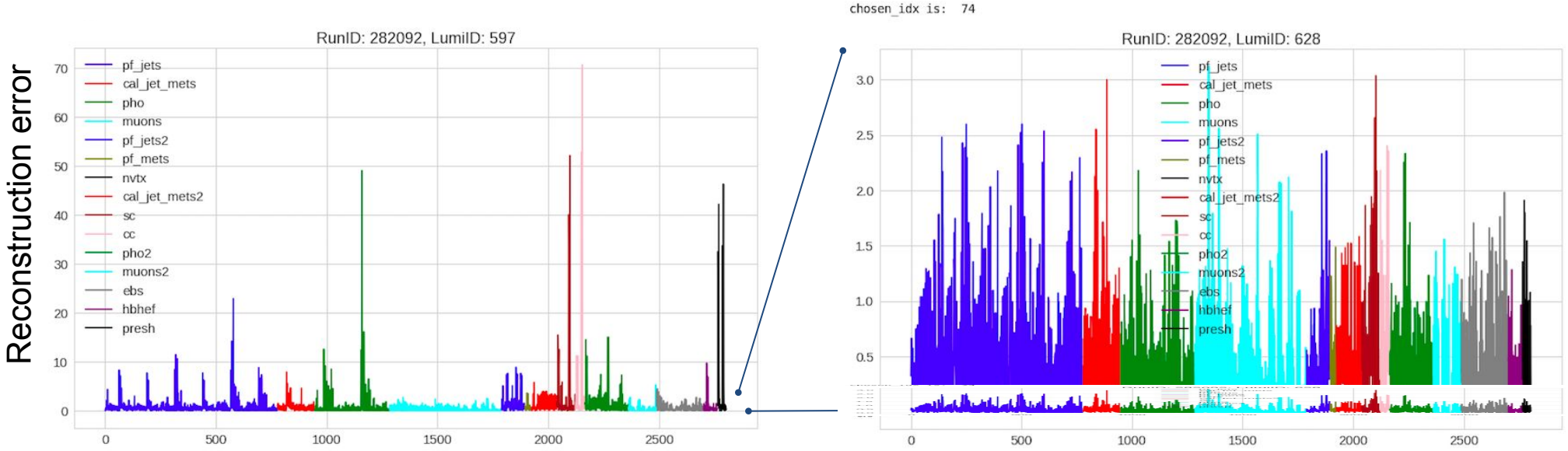
Use JetHT PD **2016** and all ~2800 features

Metric:
Good instance should have a small reconstruction error
Bad instance should have a big reconstruction error, because it is different from that seen during model training, training done on good LS
→ intuitive interpretability of the results

AE results

Performance of the model: ROC AUC score = 0.97

Reconstruction error distribution for each feature, at a fix lumisection



Features for one “bad” lumisection

Features for one “good” lumisection

Interpretation:

peaks of higher reconstructed errors for anomalous LS

uniform distribution of reconstruction errors for good LS

First tests in 2018:

AE’s decision will be feedbacked to human experts for an informed double check

Conclusions

We are looking toward automated Data Quality Assessment
to increase accuracy of decision, to limit human power and time delay

Results:

- Detecting anomalies in data is a good task for the AI
- Joined effort between CMS and Yandex/HSE groups to develop corresponding techniques applicable to the CMS use case
  - cooperation under umbrella of the CERN Open Lab
- 3-class classification approach is shown to work well in Run 1 conditions
  - up to 80% of manual work could be saved
- Special NN architecture allows to decompose anomalies by different sources without explicit training
- Much improved data quality in Run 2 requires different approach
  - 1-class classification is more appropriate
    - encouraging results using AutoEncoders

Outlook:

- Extend all studies to 2017 data, and address possible differences
- Aim for deployment proved to work elements of this system into 2018 DC workflow
  - Profit!

CERN openlab Yandex CMS

Questions?
Contact me: azzolini@cern.ch
Or the e-group: cms-ml4dc@cern.ch

backup

Data Access Policies

This is not straightforward for cooperation beyond collaboration:

◊ to be useful, the system needs access to data in real time.

◊ collaboration restricts access to physics data during grace period

◊ Yandex is not a member of CMS collaboration

Practical solution:

◊ CMS members of the team provide data processing and collecting statistics over periods of data taking

  ◊ collected data contain only integrated information, no information from individual events

◊ Yandex members of the team develop classification algorithms based on these integrated features

Channel decomposition results

Input:
2010B CMS data
separated between channels inputs

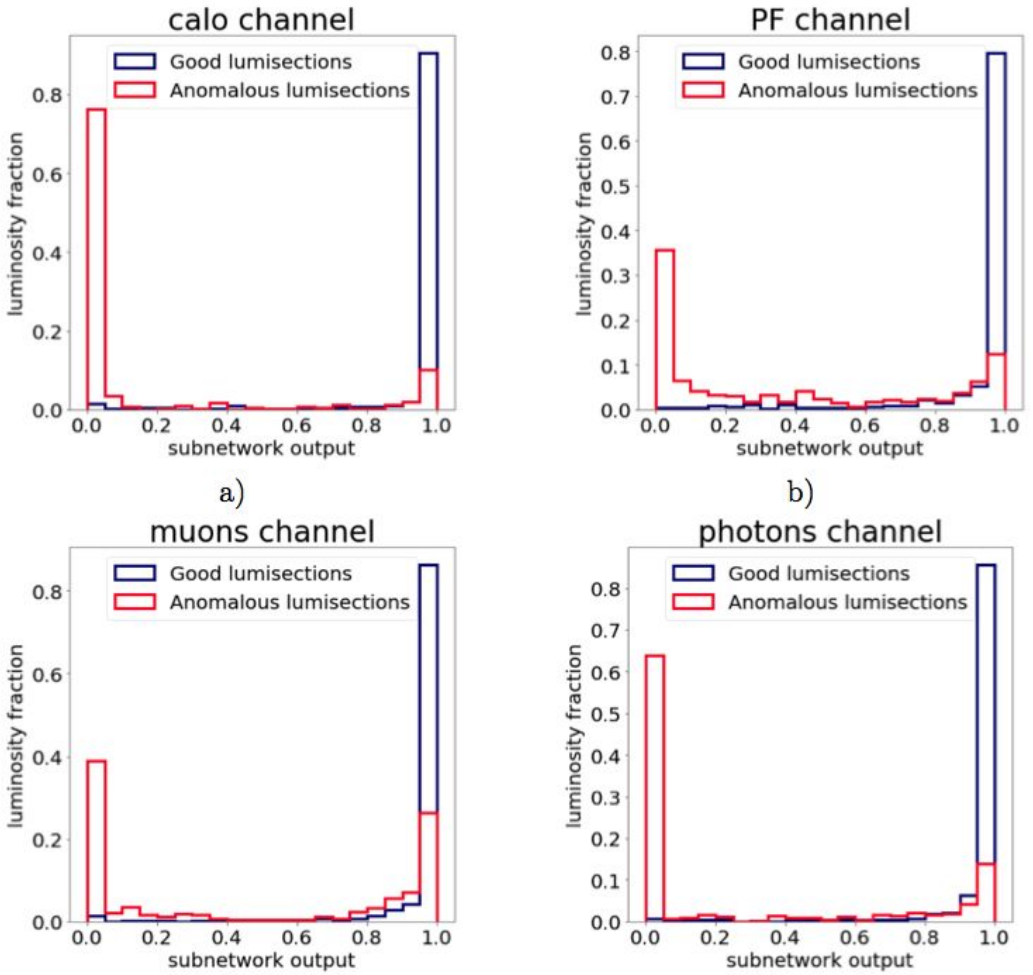
Interpretation:
◊ 'good' samples scores close to 1.

◊ 'bad' data has two options:
.visible from channel (score ~ 0)

.not visible (score ~ 1)
this data is not affected by an anomaly→ still useful for physical analysis

Global predictive power of the whole NN:
ROC AUC score = 0.96

Distributions of predictions in each NN branch.



Yandex

Russian leading search engine company having it seat at Moscow, Russia

Yandex in Wikipedia

Yandex

Type	Public company
Traded as	NASDAQ: YNDX, MCX: YNDX
Founded	1997; 19 years ago (Yandex search launched by CompTek) 2000 (Yandex company founded)
Headquarters	Lev Tolstoy st. 16, Moscow, 119021, Russia
Area served	Russia, Ukraine, Kazakhstan, Belarus and Turkey
Founder(s)	Arkady Volozh Arkady Borkovsky Ilya Segalovich
Key people	Arkady Volozh, CEO
Industry	Internet

Industry	Internet
Products	Yandex Search Yandex Direct Yandex Mail Yandex Fotki Yandex Browser Yandex Maps Yandex News Yandex Music Yandex Video Yandex Money Yandex.Translate Yandex Catalog Yandex Taxi Ya.ru Moikrug

Revenue	▲ 39.5 billion rub. (2013)[1]
Operating income	▲ 12.8 billion rub. (2013)[1]
Net income	▲ 13.5 billion rub. (2013)[1]
Total assets	▲ 34.07 billion rub. (2011)[2]
Total equity	▲ 28.95 billion rub. (2011)[2]
Employees	5,603 (*March 2015*)[3]
Slogan(s)	Yandex. Everything will be found.
Website	www.yandex.ru www.yandex.ua (Ukrainian version) www.yandex.com (English/Indonesian version) www.yandex.com.tr (Turkish version) www.yandex.kz (Kazakh version) www.yandex.by (Belarusian version)
Alexa rank	▼ 21 (February 2016)[4]

IT resources: Yandex ~ 10% × Google

Thanks to Fedor.Ratnikov@cern.ch- Yandex School of Data Analysis

Yandex in HEP

- Member of CERN OpenLab
- Member of LHCb Collaboration
  - trigger
  - B-tagging
  - monitoring
  - anomalies detection
  - computing resources
- Member of SHIP Collaboration
  - detector optimisation
  - computing resources
- Cooperating with CMS Collaboration
  - data certification
- Cooperating with ATLAS Collaboration
  - GRID optimisation
- Contributing to other Particle Physics experiments beyond CERN

Thanks to Fedor.Ratnikov@cern.ch- Yandex School of Data Analysis

Limits of a Human-based DC

. Volume budget

Limited amount of quantities that a human can process in a finite time interval

. Time delay

Offline: reconstruction data time +human intervention = ~ 1 week → Need intermediate step

. Expensive, in terms of human resources

Offline: duplication of effort (many detector and physics object experts) on weekly basis

. Makes assumptions on potential failure scenarios

DC paradigm: scrutiny of a large, but pre-def, # of histograms in comparison with a reference

Conservative strategy that could prevent unforeseen anomaly detection

. missed time dimension

Granularity of certification is lumi-section* to reduce data lost due to short pbl condition, but evaluation relies often on integrated quantities, punctual anomaly may not surface

Good news is the current system works

but volume of data has grown so large it is becoming increasingly difficult to QA all data

We aim to incorporate modern ML techniques to perform quality in future intelligent archives

*minimum time interval within a run

Automated Data Quality Assessment

2 possible approaches:

- robots like anomaly detection

  Routinely monitor same measured or reconstructed properties – “DPG”
  Rely on set of statistics and rules, and automatically state normal vs abnormal behaviors

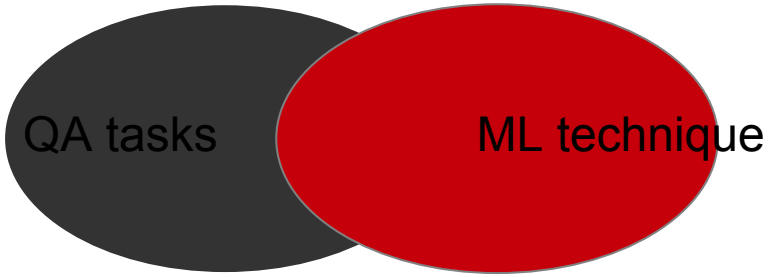
  Pro: Immediate benefit of save human intervention
  Pro: 1 to 1 verification
  Con: Constructing these statistics requires an exhaustive knowledge of the detector and all possible anomalies
  Con: no easily scalability with data volumes and detectors configurations changement

- Machine Learning-based Automated Anomaly Detection

  System is based on the measured or reconstructed physical properties – “POG/PAG”
  Step one: assist Data Quality managers by filtering most obvious cases
  Step two: anomaly detection and indication of the main feature offender

  Pro: statistics learned directly from data --> possibility of automated detection of anomaly
  Pro: adaptable to different experimental setups (including changes in the detector)
  Pro: not orthogonal to expert statistics approaches, expert statistics injection into feature set is a starting point for improvement of the system
  Con: commissioning time

Technical characteristics required for QA



WE NEED:

- operate effectively with minimal human guidance
- anticipate important events
- adapt behavior in response to changes in data content, user needs, or available resources

WE HOPE:

Developed machine learning and analytics-based solutions will:

- improve the accuracy of data quality
- establish new data quality rules to sharpen data error detection and correction
- increase the speed at which this is achieved

Technical characteristics required for QA

To do this the “intelligent data archive” will need to

- learn from its own experience

  recognize data quality problems solely from its experience with past data,
  rather than having to be told explicitly the rules for recognizing such problems.
  E.g. flag suspect data by recognizing departures from past norms
  E.g. categorize data based on the type or severity of data quality problem.

- recognize hidden patterns in incoming data streams, could learn to recognize problems

  either from explicit examples or simply its own observation of different types of data

- data access requests, ability of an archive to respond automatically to data quality problems. E.g. significant increases, in the amount of data flagged as bad or missing,

  might indicate that the data are exceeding the bounds expected by science QA algorithms
  The intelligent archive could:
  - notify DQ experts so that the issue can be further examined and resolved.
  - archive could retrieve old data to confirm a data quality problem,
    obtain data from an alternate source, or
    request that the data be recollected or reprocessed in response to confirmed DQ pbl

    (from automatic QA → to autonomous data QA)

- Fast and efficient operation

  Some current DQ applications require data to be delivered < 1h hour so the need to perform the DQ of a relatively large amount of data within a few minutes.

  Machine Learning algos are compute intensive,but we could still meet this requirement the computational effort is associated to the deriving rules or training the system;

  the rules themselves are generally computational easy to apply in an operational mode.

  If QA is a function of applying the rules, rather than deriving the rules, then the computational complexity associated with deriving the rules is not an issue, because this can be done on a subset of the data in an offline process.

Access to CMS 2016 data preparation process

2016 data workflow:

AOD ⟶ Root tree files ⟶ NumPy array ⟶ .hdf5 files

1) **Root tree** data acquisition
   Extract via a C++ CMSSW analyzer all the “interesting” variables and
   Save them in a simple Root tree files
   Feature engineering already done in the analyzer (IDs, compute quantiles of pt, rms, σ,eta, phi, …)

2) Conversion into NumPy arrays
   Easy using RootPy library
   Compatibility with SciPy for numerical integration and optimization,
   matplotlib for plotting, pandas for data analysis,
   statsmodels for statistical modeling,
   sklearn or keras for machine learning
   quick exploratory analysis in a Jupyter notebook

3) Use **pickle** to convert a python object into a character stream
   - character stream contains all the information necessary to reco the object in another python script
   - great for storing python objects (NumPy arrays)

4) Store hdf5 files in a common shared CMS and Yandex area, CERNBox

2016 Data preprocessing

Analyzer that computes 5 **quantiles, mean and rms** (7 features for each variable) from **distributions of variables** (pt, eta, phi, …) of **physical objects** (jets, photons, muons, electrons, …) from AOD 2016.

It creates numpy arrays, that can be used for training/testing.

Produces **datasets for all 20 PDs** (JetHT, SingleMuon, MuOnia, DoubleMuon, DoubleMuonLowMass,)

Each lumisection for all PDs is currently described by **2803 features**. Features other than described above are instantaneous luminosity, pathrates, cross-section.

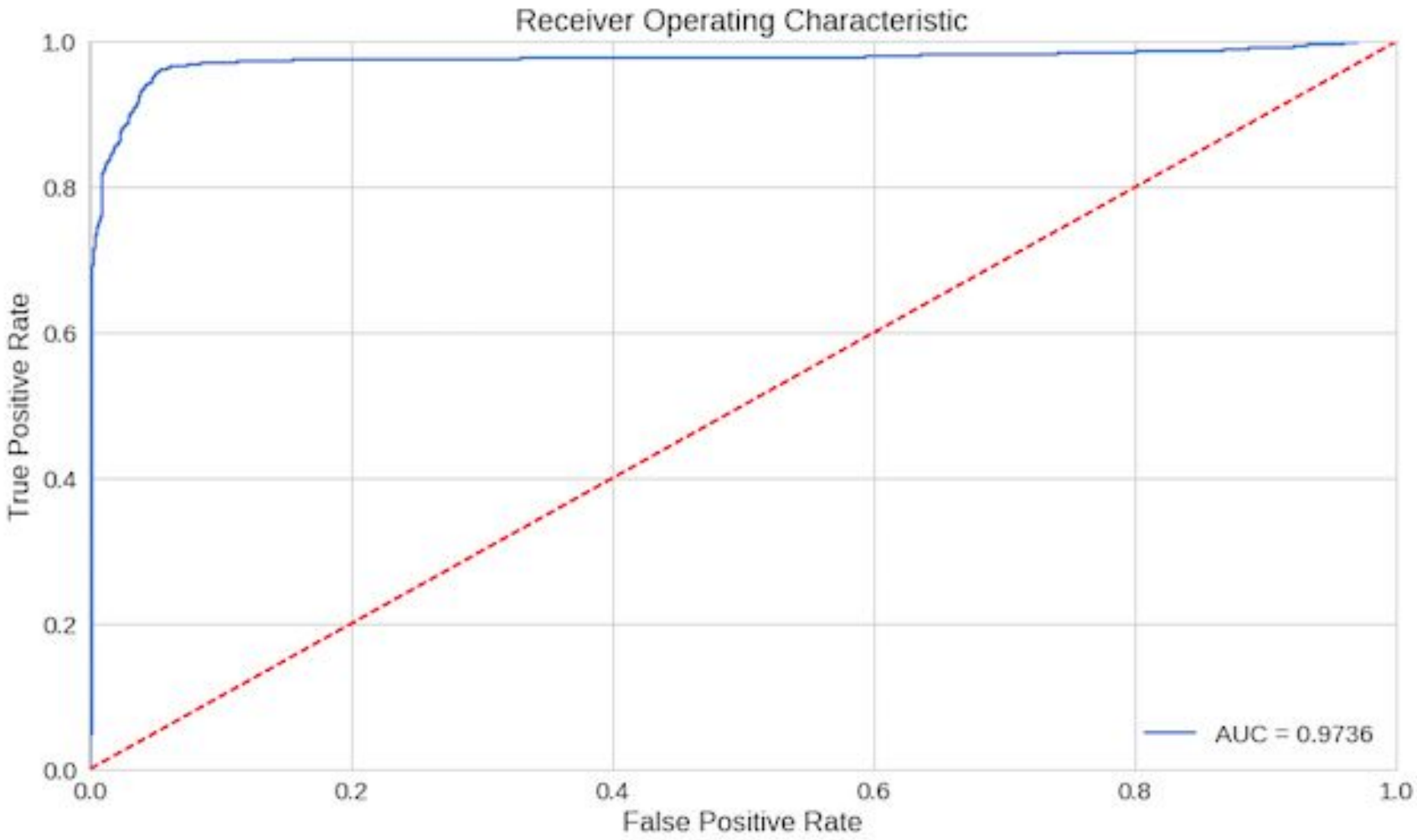
Anomalies happen relatively rarely (~2 %), are unpredictable and sparse, making it difficult to use classical supervised classification methods such as feedforward neural networks (one cannot train on anomalies well enough).

AE architecture

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 2803)	0
dense_1 (Dense)	(None, 2000)	5608000
dense_2 (Dense)	(None, 1000)	2001000
dense_3 (Dense)	(None, 500)	500500
dense_4 (Dense)	(None, 1000)	501000
dense_5 (Dense)	(None, 2000)	2002000
dense_6 (Dense)	(None, 2803)	5608803

Total params: 16,221,303
Trainable params: 16,221,303
Non-trainable params: 0

AE: ROC curve


Receiver Operating Characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
AUC = 0.9736

AE decision: interpretability

For any chosen lumisection in the test set, we can plot the absolute value of difference between actual values of features for that chosen lumisection vs reconstructed values for that lumisection, for all features. We get an reconstruction error distribution. We can group features into objects and visualize.

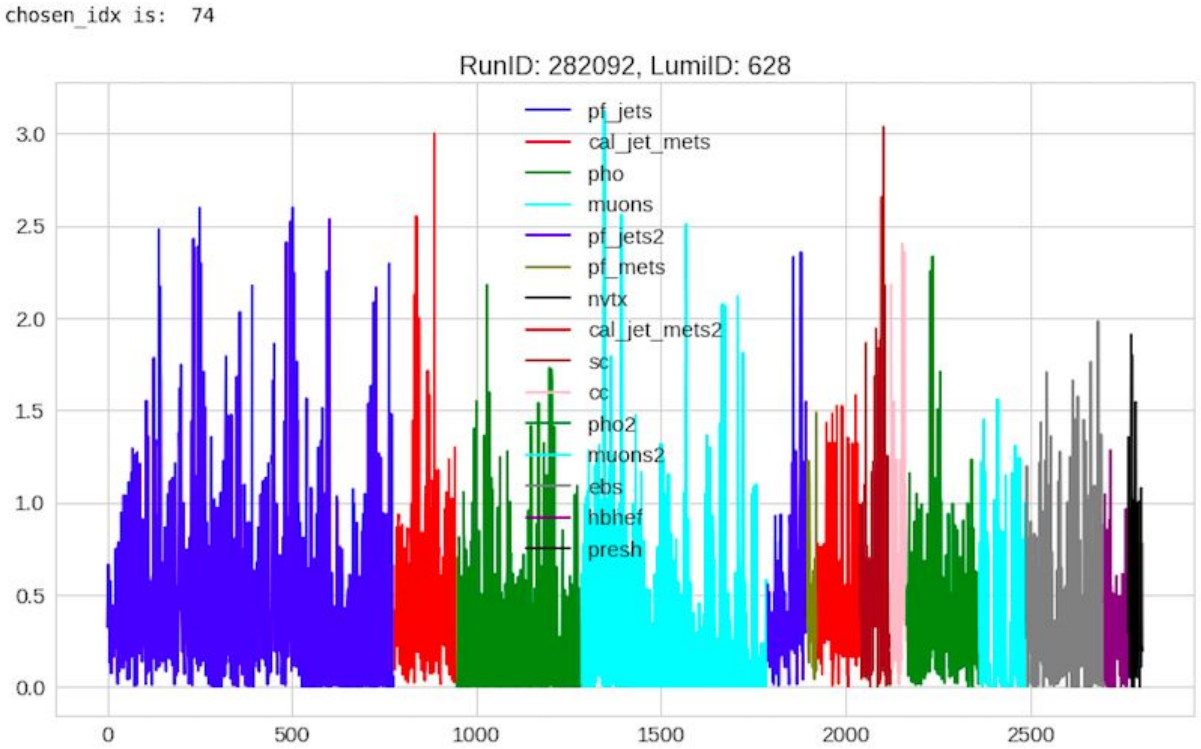
We would expect to see more or less uniform distribution for normal lumis and higher overall rec. error with peaks for anomalies. Furthermore, based on which features caused this peak(s), we can automatically assign such failure to particular subdetector - (next step, yet to be implemented).

26

AE's decision: leading the certification

Display : Reconstruction error distribution* for each feature, at a fix lumisection

Interpretability:
uniform distribution for good LS
peak of higher reconstructed error for anomalous LS



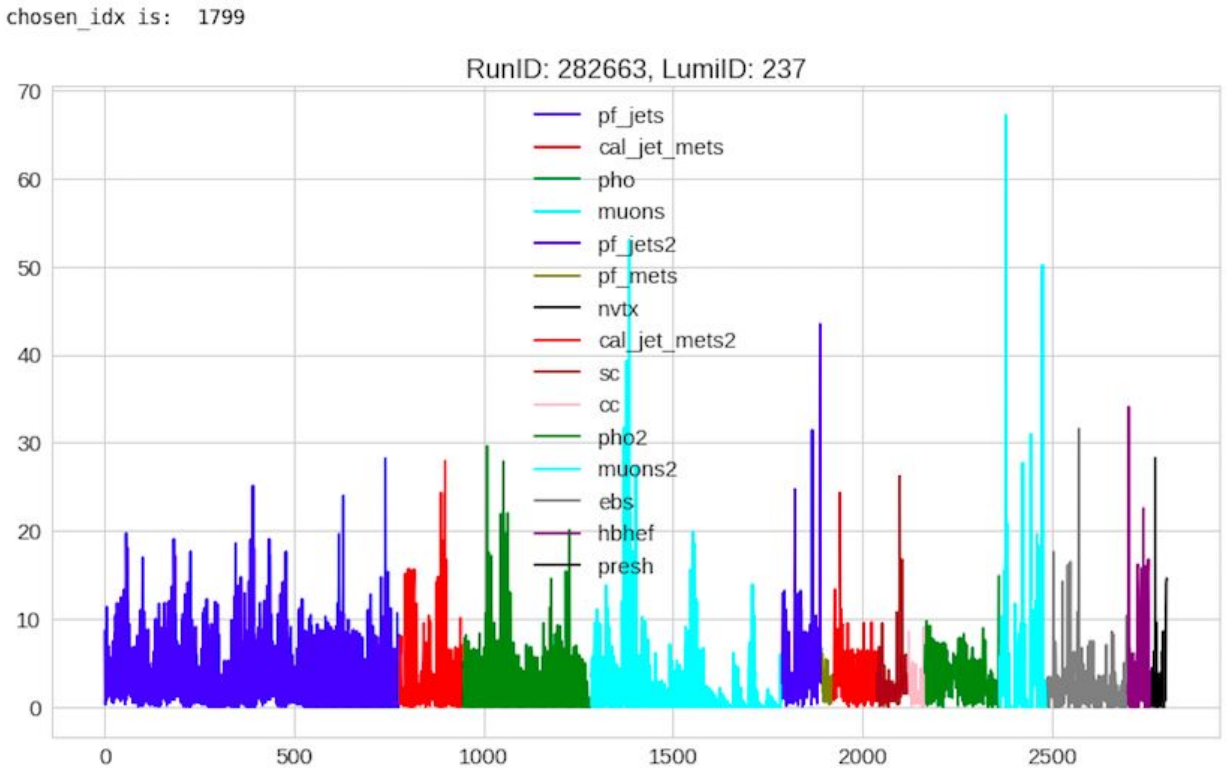
normal LS example (good LS recognized as good)

* | actual value feature - reconstructed value |

AE's decision: leading the certification

Display : Reconstruction error distribution* for each feature, at a fix lumisection

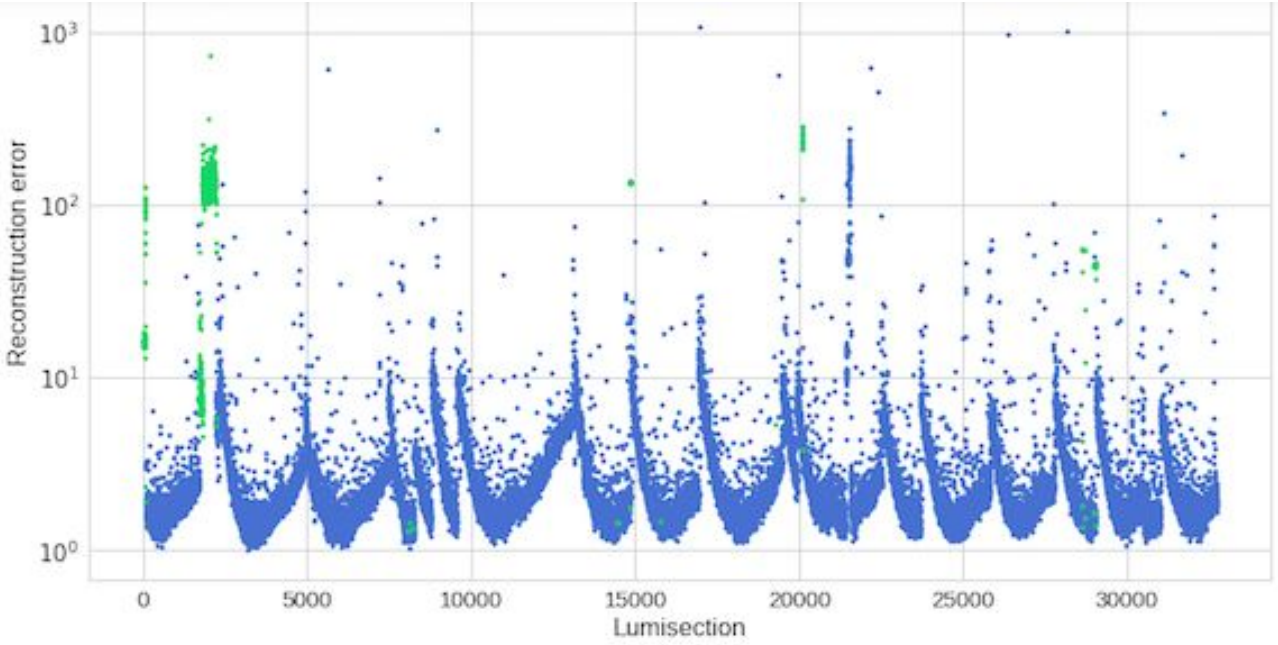
Interpretability:
uniform distribution for good LS
peak of higher reconstructed error for anomalous LS



Low PileUp lumisection (golden JSON: bad, perPDJSON: bad, CMS WBM: good, classifier: bad)

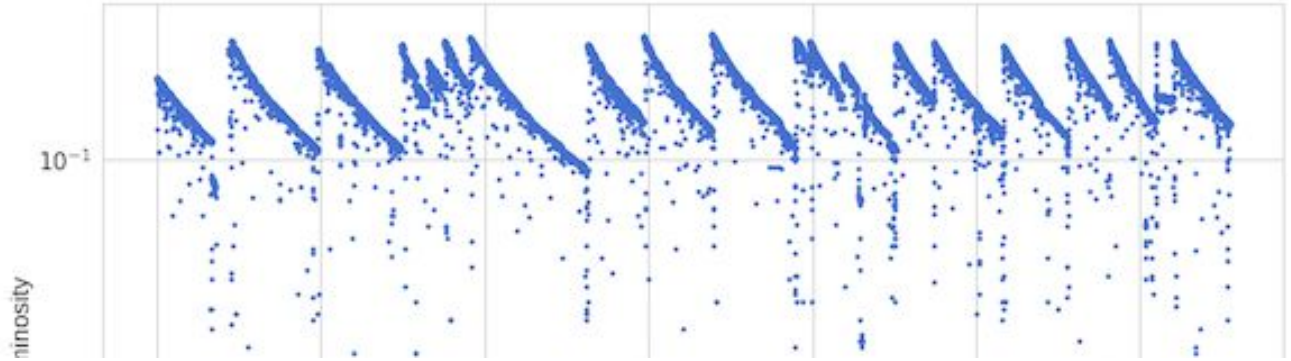
* | actual value feature - reconstructed value |

Instantaneous luminosity vs. overall rec. error




```
In [35]: plt.plot(lumisections.values, marker='o', ms=3.5, linestyle='')
         plt.ylabel('Luminosity')
         plt.yscale('log')
         plt.show()

         print(np.average(lumisections))
```



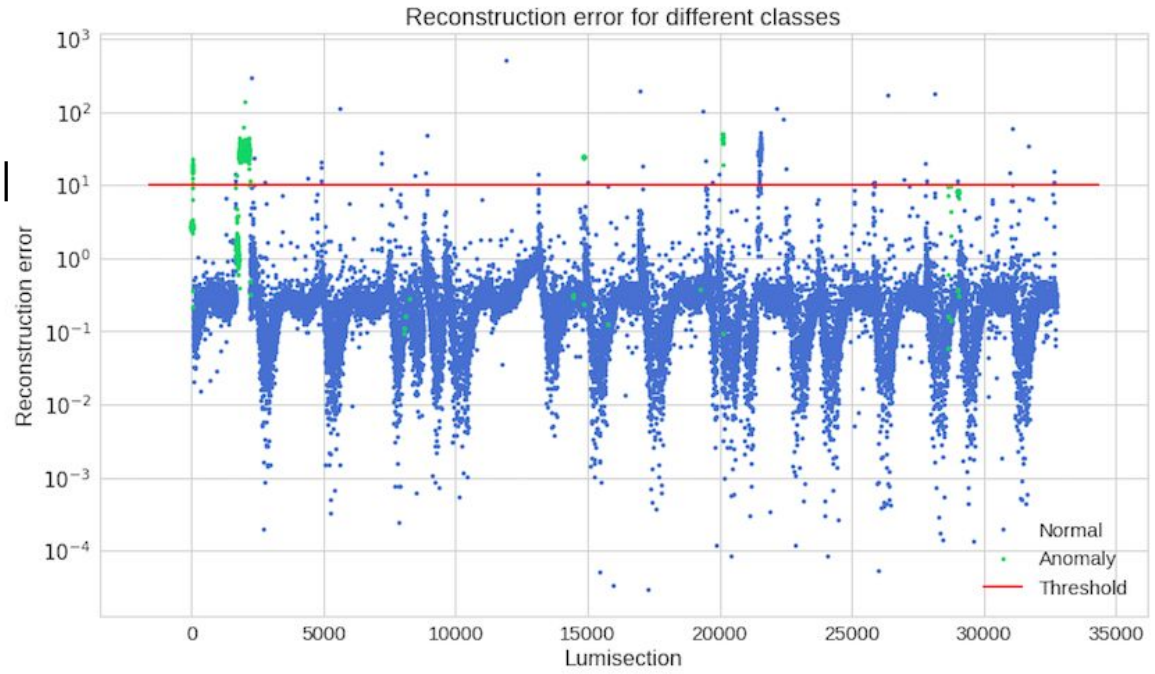
Same data like before, this time y axis has log scale

Luminosity rec. error dependency problem

Generally three ideas what to try:

1, Normalize input data with respect to their inst. luminosity before training/testing.
2, Insert luminosity feature deeper into the network to make it more important (seems most promising, because it would stay the most robust).
3, Modify metric for calculating overall reconstruction error with respect to luminosity as a special feature (plot here) ROC_AUC from 0.9736 to 0.9812, but the wavy structure is still there.

RecoErr = RecoErr -
8*|luminosity - mean(luminosity)|



Performance of different metrics

1, use mean of all ~2800 reconstruction errors for each lumisection
2, use mean of top *n* rec. errors for each lumisection
3 take mean of worst reconstructed object only

JetHT:
1, 0.9736
2, **0.9742 (*n* = 100),** 0.9736 (*n* = 500)
3, 0.9712

3PDs: (caveat, lumis missing)
1, 0.9683
2, 0.9670 (*n* = 100), **0.9685 (*n* = 500)**
3, 0.9588

We see that it doesn't make much difference, but we should probably use top *n* method because there can be situations where only one object is badly reconstructed and all others are fine. And if we used 1, method, the majority of well reconstructed objects would drag the mean down and make it think it's good.