0. Agenda

1. Motivation
2. Energy-Extraction Switches – 600A
3. Overall Architecture
4. Data Acquisition
   1. First Approach
   2. Deep Dive into the Ramp-Cycles
   3. Next Solution
5. Machine Learning
   1. Artificial Data
   2. Neural Networks
   3. Classifier
6. Outlook
7. Lessons learned
1. Motivation

- LHC is a highly sophisticated and complex machine consisting of a plentitude of systems and subsystems
- The malfunction of a single subsystem can lead to significant downtime or even severe damage
- Machine protection is an essential component to ensure smooth and failsafe operation.
- Machine Learning is a hot topic being discussed as panacea for all kinds of topics from ingrown toenails to rocket science

→ How can CERN and LHC profit of the toolbox Machine Learning best?
1. Motivation

Data is the new oil, but information is the new gold!

- Great Hunger for data to learn on!
  - Data is not information!
  - Balance of positive and negative case data essential
  - Huge amount of data useful

- LHC is a quasi-stable system
  → only a countable number of incidents/events available

Selection of the system of interest

→ Look on a system/subsystem with at least a few faulty cases
→ Energy-Extraction-Switches 600A as system under test
2. Energy-Extraction-Switches – 600A

- Energy-Extraction-Switches protect most of the circuits in case of a magnet quench
- 600A EE-Switches are AC-switches controlled by the QPS (default closed – only opened in case of an incident)
- Main parameters I_MEAS and U_DUMP_RES

![Generic circuit diagram of 600-A circuits and QPS signals.](LHC-MPP-HCP-0003-5-3_EE600A)
2. Energy-Extraction-Switches – 600A

- Each switch consists of three parallel and coupled switches/ phases
- QPS fires two switches (breaker A/B) simultaneously in case of an incident
- Breaker Z only gets fired if A&B fail!
- “Healthy” switch resistance \( \leq 550\mu\Omega \)

G.J. Coelingh et. al., 2008 IEEE International Power Modulators and High-Voltage Conference
2. Energy-Extraction-Switches – 600A

- Defective EE-Switch resistance curve
  - Switch RCS.A56.B2
  - Starttime 18.03.15
  - Endtime 12.08.15
  - Fault discovered resp. fixed after 9 weeks
  - Curve created manually by Reiner Denz using $R = \frac{U}{I}$ from Ramp-Cycles
  - Regime 1: one switch phase lost
  - Regime 2: two switch phases lost

Diagram:
- RCS.A56B2: ±319.9 A (30 points each)
- After repair!
2. Energy-Extraction-Switches – 600A

- Hierarchie and nomenclature of the relevant circuits:
  - RCD.[A12...A81][B1/2] → 16
  - RCS.[A12...A81][B1/2] → 16
  - ROD.[A12...A81][B1/2] → 16
  - ROF.[A12...A81][B1/2] → 16 → 32
  - RQ6.[L3/L7/R3/R7][B1/2] → 8
  - RQS.[A12...A81][B2/B1_alt] → 8
  - RQTD.[A12...A81][B1/2] → 16
  - RQTF.[A12...A81][B1/2] → 16 → 32
  - RQTL9.[L3/L7/R3/R7][B1/2] → 8
  - RSD[1/2].[A12...A81][B1/2] → 32
  - RSF[1/2].[A12...A81][B1/2] → 32 → 64
  - RSS.[A12...A81][B1/2] → 16
  - RU.[L4/R4] → 2
  - Total sum of EE-Switches (times 3) → 202 (606)

https://twiki.cern.ch/twiki/bin/viewauth/MP3/HWCProceduresInfo

1/23/2020
Andreas Müller
Extracting Rules Parameters for CalsDB:

- Example circuit: RCD.A12B1
- Find current in circuit-table:
  \[ \text{PC} = \text{RPMBB.UA23.RCD.A12B1} \]
  \[ \text{I} = \text{RPMBB.UA23.RCD.A12B1:}_\text{I\_MEAS} \]
- Find EESwitch in Switch table
  \[ \text{RCD.A12B1} \quad \text{gives} \quad \text{EE} = \text{UA23.RCD.A12B1} \]
  \[ \text{U} = \text{DQEMC.UA23.RCD.A12B1:}_\text{U\_DUMP\_RES} \]

But – take care, NxCals has different access parameters…
3. Overall Architecture of the Project

- Database Cals (or alternatively NxCals)
- Python
- Jupyter on SWAN
- Evaluate LHC Run 2
- Suggest Monitoring/Alarming Process
4.1 Data Acquisition – LHC Beam Modes

- Exemplary Scan:
  - Starttime: 2016-02-09 07:40:00 - Endtime: 2016-03-15 01:00:00 – duration ~ 35 days
4.1 Data Acquisition – Ramp Cycles

- Exemplary Scan:
  - Starttime: 2016-02-09 07:40:00
  - Endtime: 2016-03-15 01:00:00
  - Duration: 34 days, 17:20:00

- Ramp Cycle Criterion:
  - $310A < |I_{MEAS_{Peak}}|
  - Drop others
4.1 Data Acquisition – Ramp Cycles

Acquisition Process

• Collect all beam mode sequences 21 → 2 (Nobeam → Setup) → Dataframe with beam mode time-periods (circuit-independent)

• Scan I_MEAS in beam mode sequence for peak values → Calculate time-slot for full ramp-cycle → Dataframe with ramp-cycle time-periods (circuit-dependent)

• Scan I_MEAS and U_DUMP_RES for all ramp-cycles → Parameters are stored differently! → Rescale I_MEAS and U_DUMP_RES → Calculate resistance series
4.1 Data Acquisition – Resistance calculation

\[ R = 365.92 \, \mu \Omega \]
4.1 Data Acquisition – Resistance calculation

- Resistance healthy regime $\sim 400 \, \mu \Omega$

06_CalculateResistanceSynchronized.ipynb
4.1 Data Acquisition – Resistance series

- Resistance faulty regime \( \sim 1150 \ \mu\Omega \)

08_CalculateResistanceForOneSwitch.ipynb
4.1 Data Acquisition – Resistance series

\[
x = \text{manually evaluated points}
\]
4.1 Data Acquisition – Ramp Cycles

• But, there were some strange phenomena, depending on the time raster of the resistance data.
• One scan took about 10 hours for a step-size of 2

→ Deep Dive into Data before starting a long-running multi-process job…
4.2 Deep Dive – Special Ramp Cycles Ia
4.2 Deep Dive – Special Ramp Cycles IIa
4.2 Deep Dive – Special Ramp Cycles Ib
4.2 Deep Dive – Normal Ramp Cycles IIb
4.3 Data Acquisition – KI Rules

Set of Rules to identify a complete Max → Min-Cycle

1. Only beam mode 19 → next beam-mode
   → true ramp-cycles, no commissioning etc.

2. Threshold
   \( I_{\text{MEAS.max}}() > 290 \, \text{A} \) and \( I_{\text{MEAS.min}}() < -290 \, \text{A} \)
   → sufficient ramp level
   (Ramp current depends on power converter!)

3. Max-Min-Sequence
   Maximum is followed by a Minimum (or vice versa)
   → don’t detect single peaks

4. Symmetricity
   \( |I_{\text{MEAS.max}}() + I_{\text{MEAS.min}}()| < 10 \, \text{A} \)
   → symmetric cycle

5. Scan Duration
   \( I_{\text{MEAS.max}}().\text{time} \rightarrow I_{\text{MEAS.min}}().\text{time} \sim 445.5 \, \text{seconds} \)
   → continuous cycle without gaps
4.3 Data Acquisition – KI Rules

- “Reference” ramp-cycle
5.1 Artificial Data – Generation

- **Generated artificial “EE-Switch” data**
  - Generate baseline – “healthy” switch
    - Normal distribution of events on a baseline (mean) of $\mu = 500 \mu\Omega$ with added noise $\sigma = 25 \mu\Omega$
  - Generate events
    - Class 0, 1, 2, 3 – “healthy” switch (no event)
    - Class 4, 5 – only precursor (1 phase)
    - Class 6 – precursor with full peak (1, 2 phases)
    - Class 7 – only full peak (2 phases)

![Graph showing event frequency distribution](11_create_artificial_switch_data_jfl.ipynb)

```
Name: event, dtype: int64
```

![Histogram of artificial data](12a_create_artificial_data_new.ipynb)
5.1 Artificial Data – Generation

Each window consists of 10 time series with artificial resistance data with 2,000 points in time.
5.1 Artificial Data – General

- Confusion Matrix (CM as a measure of accuracy)
  - Used convention as in `sklearn.metrics.confusion_matrix`
  - False Positive = Type I Error, False Negative = Type II Error

<table>
<thead>
<tr>
<th>Actual condition</th>
<th>Predicted condition</th>
<th>CM Confusion Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neg</td>
<td>True negative</td>
<td>False positive Type I</td>
</tr>
<tr>
<td>Pos</td>
<td>False negative Type II</td>
<td>True positive</td>
</tr>
</tbody>
</table>

5.2 Artificial Data – Neural Networks

- **Trained Classifiers with artificial resistance data**
  - 1,000 artificial resistance time series (each 2,000 points in time)
  - Train-test split 0.2 → 800 training data, 100 validation data
  - Convolutional NN
    - Accuracy 0.445
    - Precision 0.24722222
    - Recall 0.40825688
    - Runtime 44.85 seconds
  - Multilayer-Perceptron
    - Accuracy 0.455
    - Precision 0.2275
    - Recall 0.5
    - Runtime 123.90 seconds
  - Fully Convolutional NN
    - Accuracy 1.0
    - Precision 1.0
    - Recall 1.0
    - Runtime 9179.87 seconds

Algorithms used as in: H.I. Fawaz et. al., Deep Learning for Time Series Classification
5.3 Artificial Data – Classification

- Trained Classifiers with artificial resistance data
  - 10,000 artificial resistance time series (each 1,000 points in time)
  - Train-test split 0.1 → 9,000 training data, 1,000 validation data
  - Logistic Regression Classifier
    - Precision 0.97318008
    - Recall 0.99607843
    - F1-Score 0.98449612
    - Runtime 1.73 seconds
  - K-Nearest-Neighbors Classifier
    - Precision 1.0
    - Recall 1.0
    - F1-Score 1.0
    - Runtime 0.71 seconds
  - Support Vector Machine
    - Precision 0.96346154
    - Recall 0.98235294
    - F1-Score 0.97281553
    - Runtime 12.47 seconds

<table>
<thead>
<tr>
<th>CM</th>
<th>Predicted condition</th>
<th>Actual condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logistic</td>
<td></td>
<td>Neg</td>
</tr>
<tr>
<td>Regression</td>
<td></td>
<td>Neg</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pos</td>
</tr>
<tr>
<td>K-Nearest-Neighbors</td>
<td></td>
<td>Neg</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pos</td>
</tr>
<tr>
<td>Linear</td>
<td></td>
<td>Neg</td>
</tr>
<tr>
<td>SVM</td>
<td></td>
<td>Pos</td>
</tr>
</tbody>
</table>
5.3 Artificial Data – Classification

- **Trained Classifiers with artificial resistance data**
  - 10,000 artificial resistance time series (each 1,000 points in time)
  - Train-test split 0.1 → 9,000 training data, 1,000 validation data
  - Support Vector Machine – Poly 3
    - Precision: 0.88235294
    - Recall: 1.0
    - F1-Score: 0.9375
    - Runtime: 129.6 seconds
  - Support Vector Machine – Poly 2
    - Precision: 0.99804305
    - Recall: 1.0
    - F1-Score: 0.99902057
    - Runtime: 71.3 seconds
  - Support Vector Machine
    - Precision: 1.0
    - Recall: 1.0
    - F1-Score: 1.0
    - Runtime: 4.66 seconds

<table>
<thead>
<tr>
<th>CM SVM Poly 3</th>
<th>Predicted condition</th>
<th>Neg</th>
<th>Pos</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual condition</td>
<td>Neg</td>
<td>510</td>
<td>0</td>
</tr>
<tr>
<td>Pos</td>
<td>68</td>
<td>422</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CM SVM Poly 2</th>
<th>Predicted condition</th>
<th>Neg</th>
<th>Pos</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual condition</td>
<td>Neg</td>
<td>510</td>
<td>0</td>
</tr>
<tr>
<td>Pos</td>
<td>1</td>
<td>489</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CM SVM Kernel rbf</th>
<th>Predicted condition</th>
<th>Neg</th>
<th>Pos</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual condition</td>
<td>Neg</td>
<td>510</td>
<td>0</td>
</tr>
<tr>
<td>Pos</td>
<td>0</td>
<td>490</td>
<td></td>
</tr>
</tbody>
</table>
5.3 Artificial Data – Classification

- Trained Classifiers with artificial resistance data
  - 10,000 artificial resistance time series (each 1,000 points in time)
  - Train-test split 0.1 → 9,000 training data, 1,000 validation data
  - Decision Tree Classifier
    - Precision: 0.97851562
    - Recall: 0.98235294
    - F1-Score: 0.98043053
    - Runtime: 309 seconds
  - Naïve Bayes Classifier
    - Precision: 1.0
    - Recall: 1.0
    - F1-Score: 1.0
    - Runtime: 0.23 seconds
  - Random Forest Classifier
    - Precision: 0.94434137
    - Recall: 0.99803922
    - F1-Score: 0.97044805
    - Runtime: 64.1 seconds

<table>
<thead>
<tr>
<th>CM</th>
<th>Predicted condition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Neg</td>
</tr>
<tr>
<td>Decision Tree</td>
<td>Neg</td>
</tr>
<tr>
<td></td>
<td>Pos</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CM</th>
<th>Predicted condition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Neg</td>
</tr>
<tr>
<td>Naïve Bayes</td>
<td>Neg</td>
</tr>
<tr>
<td></td>
<td>Pos</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CM</th>
<th>Predicted condition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Neg</td>
</tr>
<tr>
<td>Random Forest 2</td>
<td>Neg</td>
</tr>
<tr>
<td></td>
<td>Pos</td>
</tr>
</tbody>
</table>
5.3 Artificial Data – Classification

- Trained Classifiers with artificial resistance data
  - 10,000 artificial resistance time series (each 1,000 points in time)
  - Train-test split 0.1 → 9,000 training data, 1,000 validation data
  - Random Forest Classifier
    - Precision: 0.9921875
    - Recall: 0.99607843
    - F1-Score: 0.99412916
    - Runtime: 65.6 seconds
  - Random Forest Classifier
    - Precision: 0.99607843
    - Recall: 0.99607843
    - F1-Score: 0.99803922
    - Runtime: 97.3 seconds
  - Random Forest Classifier
    - Precision: 0.99803922
    - Recall: 0.99803922
    - F1-Score: 0.99803922
    - Runtime: 243.7 seconds

<table>
<thead>
<tr>
<th>CM</th>
<th>Predicted condition</th>
<th>Actual condition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Neg</td>
<td>Pos</td>
</tr>
<tr>
<td>Random Forest 3</td>
<td>508</td>
<td>2</td>
</tr>
<tr>
<td>Random Forest 5</td>
<td>508</td>
<td>2</td>
</tr>
<tr>
<td>Random Forest 10</td>
<td>509</td>
<td>1</td>
</tr>
</tbody>
</table>
5.3 Artificial Data – Classification

- Trained Classifiers with artificial resistance data
  - 10,000 artificial resistance time series (each 1,000 points in time)
  - Train-test split 0.1 → 9,000 training data, 1,000 validation data
  - Random Forest Classifier
    - Precision: 0.99804305
    - Recall: 1.0
    - F1-Score: 0.99902057
    - Runtime: 100.2 seconds
  - Random Forest Classifier
    - Precision: 1.0
    - Recall: 1.0
    - F1-Score: 1.0
    - Runtime: 258.4 seconds
  - Random Forest Classifier
    - Precision: 1.0
    - Recall: 1.0
    - F1-Score: 1.0
    - Runtime: 348.0 seconds

**Confusion Matrixes**

**Random Forest Classifier 12**

<table>
<thead>
<tr>
<th>Actual condition</th>
<th>Predicted condition</th>
<th>Neg</th>
<th>Pos</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neg</td>
<td></td>
<td>510</td>
<td>0</td>
</tr>
<tr>
<td>Pos</td>
<td></td>
<td>1</td>
<td>489</td>
</tr>
</tbody>
</table>

**Random Forest Classifier 13**

<table>
<thead>
<tr>
<th>Actual condition</th>
<th>Predicted condition</th>
<th>Neg</th>
<th>Pos</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neg</td>
<td></td>
<td>510</td>
<td>0</td>
</tr>
<tr>
<td>Pos</td>
<td></td>
<td>0</td>
<td>490</td>
</tr>
</tbody>
</table>

**Random Forest Classifier 15**

<table>
<thead>
<tr>
<th>Actual condition</th>
<th>Predicted condition</th>
<th>Neg</th>
<th>Pos</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neg</td>
<td></td>
<td>510</td>
<td>0</td>
</tr>
<tr>
<td>Pos</td>
<td></td>
<td>0</td>
<td>490</td>
</tr>
</tbody>
</table>
5.3 Artificial Data – Classification

- Trained winners from classifiers with less artificial resistance data
  - 1,000 artificial resistance time series (each 1,000 points in time)
  - Train-test split 0.2 → 800 training data, 200 validation data
  - K-Nearest-Neighbors Classifier
    - Precision 0.68309859
    - Recall 1.0
    - F1-Score 0.81171548
    - Runtime 0.025 seconds
  - Support Vector Machine (rbf)
    - Precision 1.0
    - Recall 1.0
    - F1-Score 1.0
    - Runtime 0.22 seconds
  - Naïve Bayes Classifier
    - Precision 1.0
    - Recall 1.0
    - F1-Score 1.0
    - Runtime 0.035 seconds

<table>
<thead>
<tr>
<th>CM</th>
<th>Predicted condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>K-Nearest-Neighbors</td>
<td></td>
</tr>
<tr>
<td>Neg</td>
<td>97</td>
</tr>
<tr>
<td>Pos</td>
<td>45</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CM</th>
<th>Predicted condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM Kernel rbf</td>
<td></td>
</tr>
<tr>
<td>Neg</td>
<td>97</td>
</tr>
<tr>
<td>Pos</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CM</th>
<th>Predicted condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naïve Bayes</td>
<td></td>
</tr>
<tr>
<td>Neg</td>
<td>97</td>
</tr>
<tr>
<td>Pos</td>
<td>0</td>
</tr>
</tbody>
</table>
5.3 Artificial Data – Classification

- Trained winners from classifiers with less artificial resistance data
  - 100 artificial resistance time series (each 1,000 points in time)
  - Train-test split 0.2 → 80 training data, 20 validation data
  - Support Vector Machine (rbf)
    - Precision 1.0
    - Recall 1.0
    - F1-Score 1.0
    - Runtime 0.0017 seconds
  - Naïve Bayes Classifier
    - Precision 0.91666667
    - Recall 1.0
    - F1-Score 0.95652174
    - Runtime 0.0017 seconds

<table>
<thead>
<tr>
<th>CM</th>
<th>Predicted condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM Kernel rbf</td>
<td>Neg</td>
</tr>
<tr>
<td>Actual condition</td>
<td>Neg</td>
</tr>
<tr>
<td></td>
<td>Pos</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CM</th>
<th>Predicted condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naïve Bayes</td>
<td>Neg</td>
</tr>
<tr>
<td>Actual condition</td>
<td>Neg</td>
</tr>
<tr>
<td></td>
<td>Pos</td>
</tr>
</tbody>
</table>
5.3 Artificial Data – Classification

- Trained winners from classifiers with less artificial resistance data
  - 50 artificial resistance time series (each 1,000 points in time)
  - Train-test split 0.2 → 40 training data, 10 validation data
  - Support Vector Machine (rbf)
    - Precision 1.0
    - Recall 1.0
    - F1-Score 1.0
    - Runtime 0.00257 seconds

- Support Vector Machine (rbf)
  - Precision 1.0
  - Recall 1.0
  - F1-Score 1.0
  - Runtime 0.00255 seconds

- Support Vector Machine (rbf)
  - Precision 1.0
  - Recall 0.0
  - F1-Score NAN
  - Runtime 0.00072

<table>
<thead>
<tr>
<th>CM SVM Kernel rbf</th>
<th>Predicted condition</th>
<th>Actual condition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Neg</td>
<td>Pos</td>
</tr>
<tr>
<td>Neg</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>Pos</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CM SVM Kernel rbf</th>
<th>Predicted condition</th>
<th>Actual condition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Neg</td>
<td>Pos</td>
</tr>
<tr>
<td>Neg</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Pos</td>
<td>0</td>
<td>7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CM SVM Kernel rbf</th>
<th>Predicted condition</th>
<th>Actual condition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Neg</td>
<td>Pos</td>
</tr>
<tr>
<td>Neg</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Pos</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
6. Outlook – Work to be done

• Data Acquisition
  • Get a proper full-blown resistance matrix
  • Use HTCondor jobs with python scripts

• Machine Learning
  • Train model with real resistance data
  • Evaluate pre-trained model with real resistance data

• Suggest continuous monitoring process for switches
  • Script to be manually executed after beam mode 19 - cycling
7. Lessons Learned

- A glimpse of “Pythonic” thinking
  - Dynamically typed language → types change e.g. from Dataframe to Series
  - Use implicit tools e.g. for iteration np.where() instead of for-loop
  - Code needs “re-reading” and “re-thinking” (at least for me…)
- SWAN platform / Jupyter notebooks good for rapid prototypes
  - Archaic debugging
  - not for big projects…
- NxCals is not faster than Cals (in this realization)
  - Nobody won a Kaggle challenge with Spark yet, but I’m convinced it will happen…
- Dataframes are immutable
- Dataframes can’t be easily plotted – conversion takes time (toPandas)
- Machine learning is a box of tools
  - Use the right tool for your problem, not the most sophisticated
  - Some problems could be solved even without machine learning, just in an old-school way…
  - Machine learning needs data! → take care in the few data limit!
- Knowledge is distributed
  - Take care of a good handover process
8. Famous last words

I am still confused, but on a higher level…