Machine Learning Tutorial: a Physics Case.

F. Neves

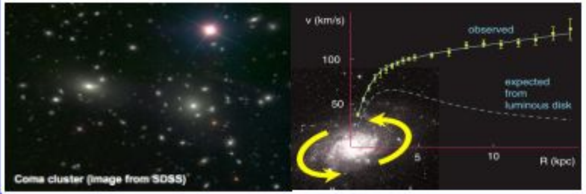
Data Science School
27-30 June, 2022

The Physics Question: Dark Matter

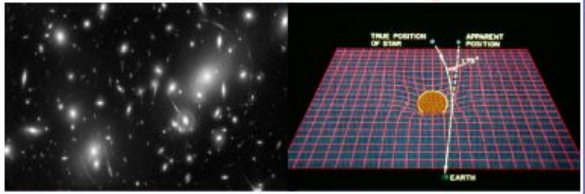
Dark Matter

Multiple scientific evidences indicate that **~85%** of the matter content of the universe is **Dark Matter (DM)**

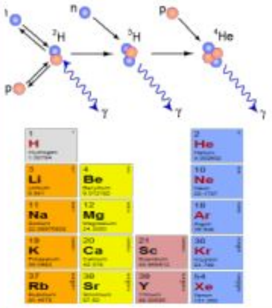
Motion of stars, gas and galaxies
(1st reference in 1933 by Fritz Zwicky)



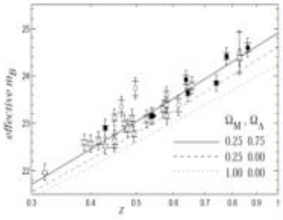
Gravitational Lensing



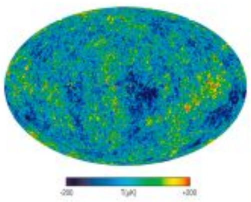
Nucleosynthesis



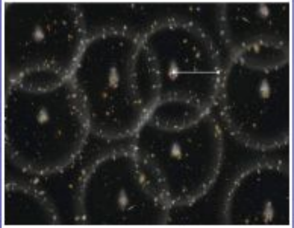
Supernovae

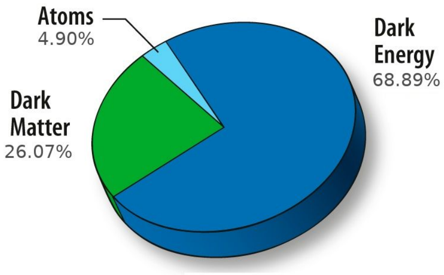


Cosmic microwave background

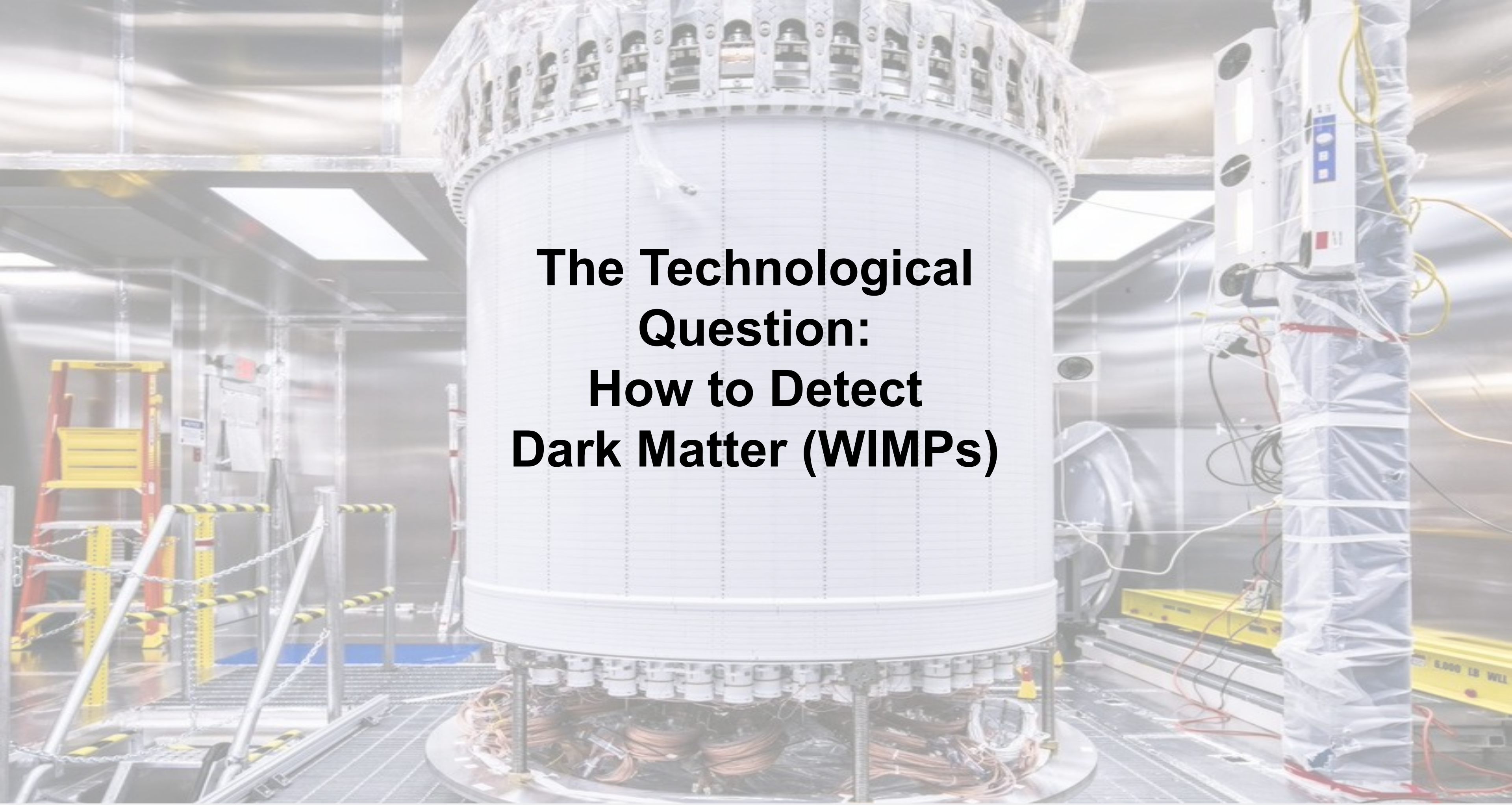


Baryonic acoustic oscillations



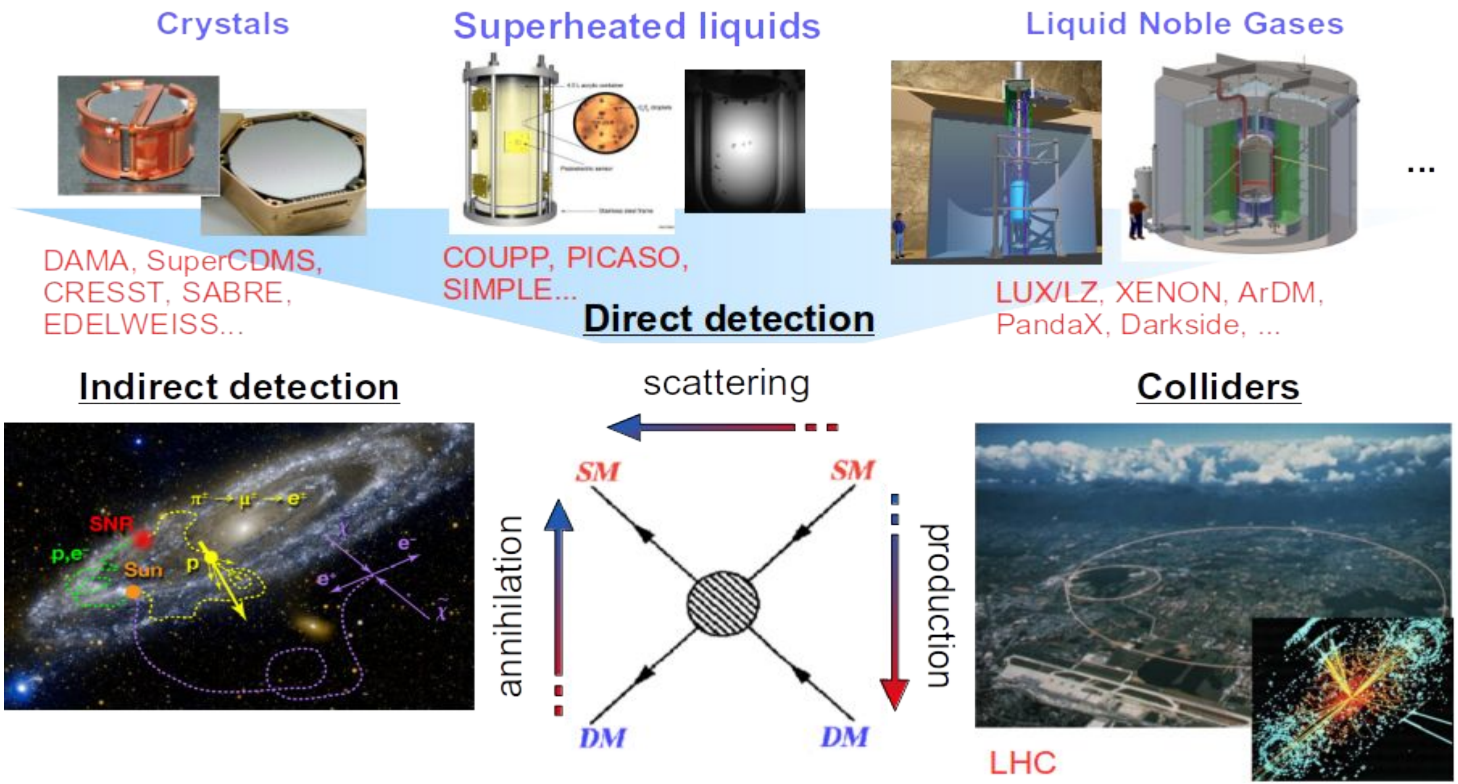


WIMPS? Axions? …
Chaplygin Gas? Neutrinos? …
Branons? Primordial Black Holes? …
…
…



The Technological Question: How to Detect Dark Matter (WIMPs)

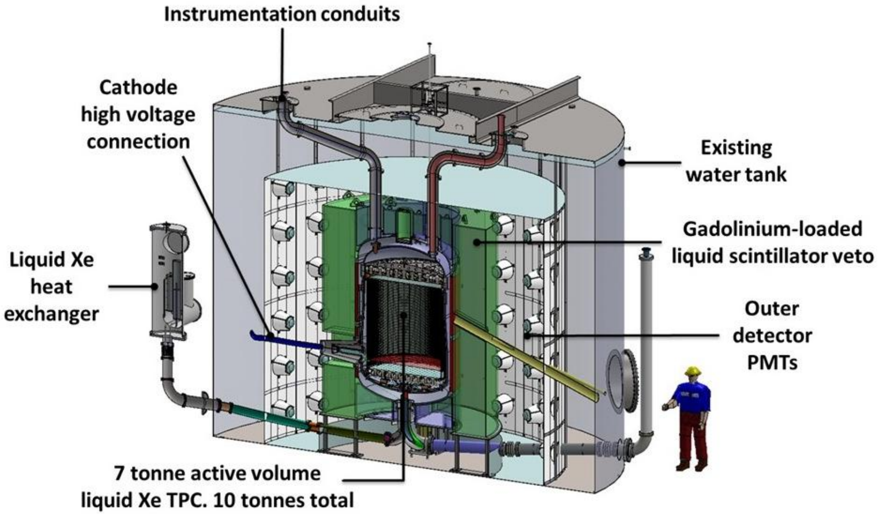
How to detect WIMPs?



The LZ Detector

LZ is the successor of **LUX**, the most sensitive DM direct detection experiment from 2013 to 2017.

- Features a **7 tonne dual-phase xenon time projection chamber** (TPC).
- Ultra-low BG environment within the detector is fitting for rare event searches:
  - Direct search of dark matter in the form of WIMPs (main goal)
  - Neutrinoless double beta decay
  - CEvNS of solar neutrinos
- Features two active veto systems:
  - LXe “Skin” layer
  - Outer detector (OD) with GdLS
- Will be operated at a depth of 1.5 km in the Sanford Underground Research Facility (SURF) in Lead, South Dakota (USA)
- **… completed its 1st science run!**

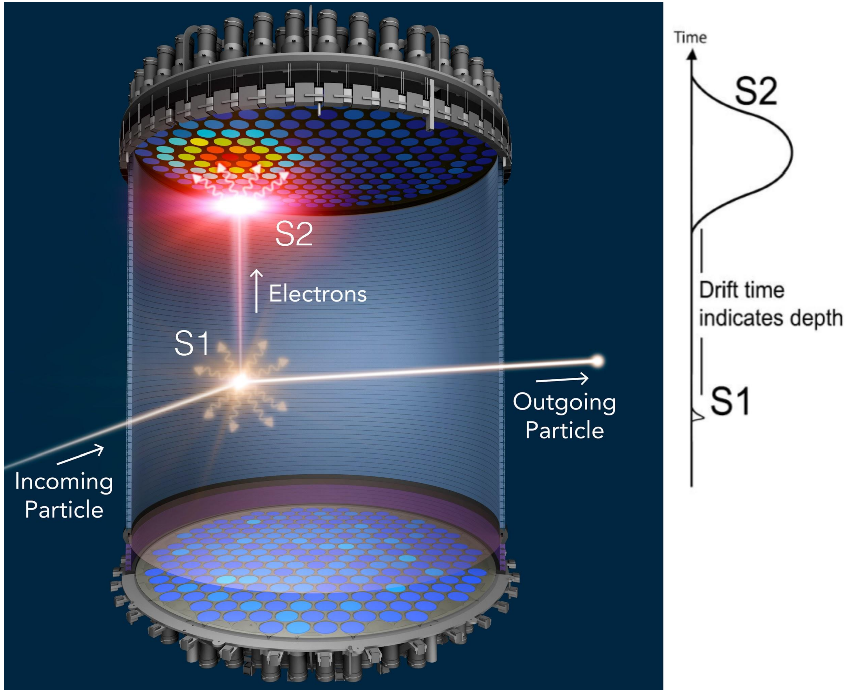


Schematic of the LZ detector,
a 7 tonne dual-phase Xenon time projection chamber (TPC)

LZ TPC Operating Principle

1. An energy deposition in the LXe produces **prompt scintillation light (S1)** and ionization electrons.
2. The electrons that do not recombine are drifted to the liquid-gas interface and extracted into the gas phase, creating **electroluminescence light (S2)**

- ★ **Deposited energy is reconstructed using both the S1 and S2 signals.**
- ★ The depth of the interaction can be obtained by the **time difference between the S1 and S2 signals.**
- ★ The XY position can be reconstructed using the **light pattern generated by the S2** signal on the top PMT array.



We get a full 3D reconstruction of the interaction!

LZ Recorded Event (the data)

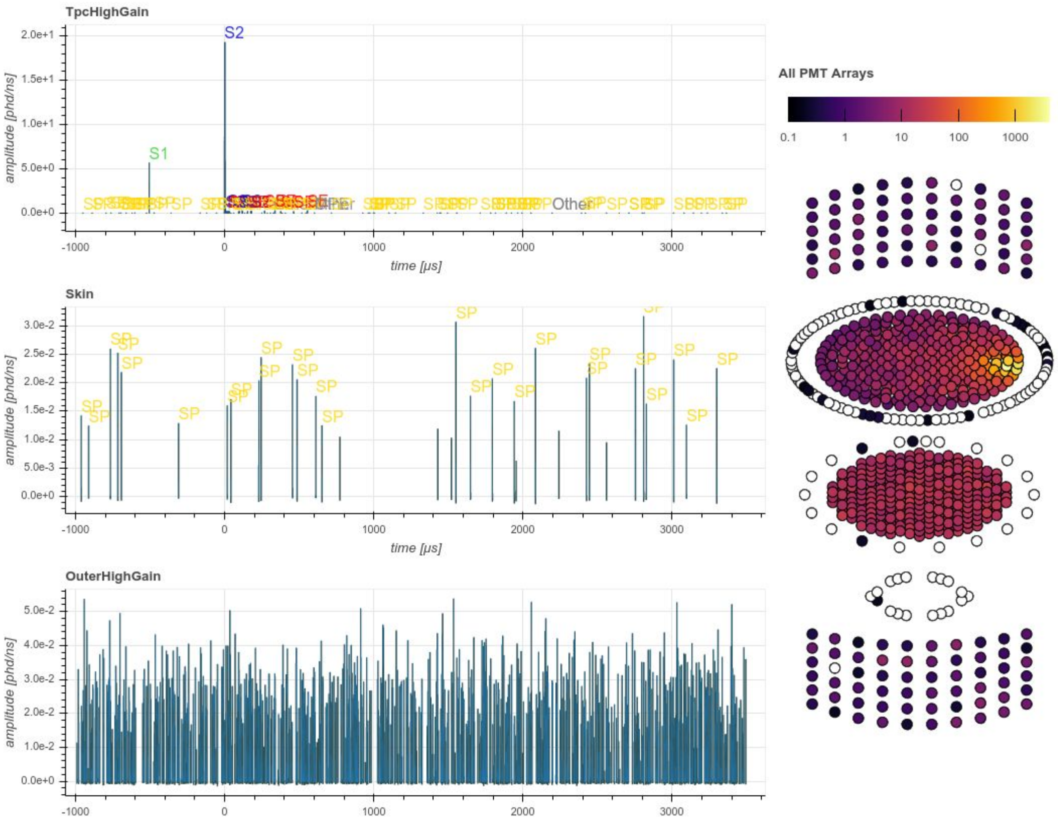
An event is recorded as a time series voltage response per PMT (channel), aka **a “timeline”**

Timelines are recorded in 2 gain modes in the TPC and OD: High Gain (HG) and Low Gain (LG); And single mode in the Skin detector.

All data presented is **simulated**, no Monte Carlo truth info is available for developers

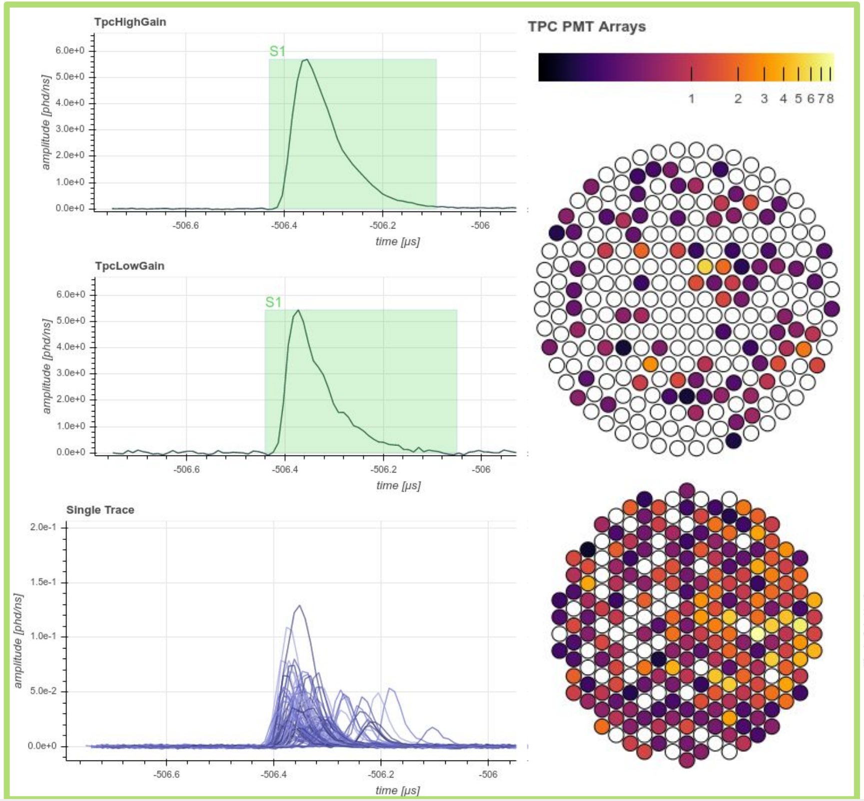
S1 and S2 pulses are required to fully describe an event. However, these are not the only pulse types in the data!

- Some originate from known detector behaviour (e.g., SE, S2 tails, e-trains, SPEs, dark counts, Afterpulsing)
- Some other are a consequence of electronic and/or analysis glitches (e.g., baselines, oversplitting, merging, etc...)

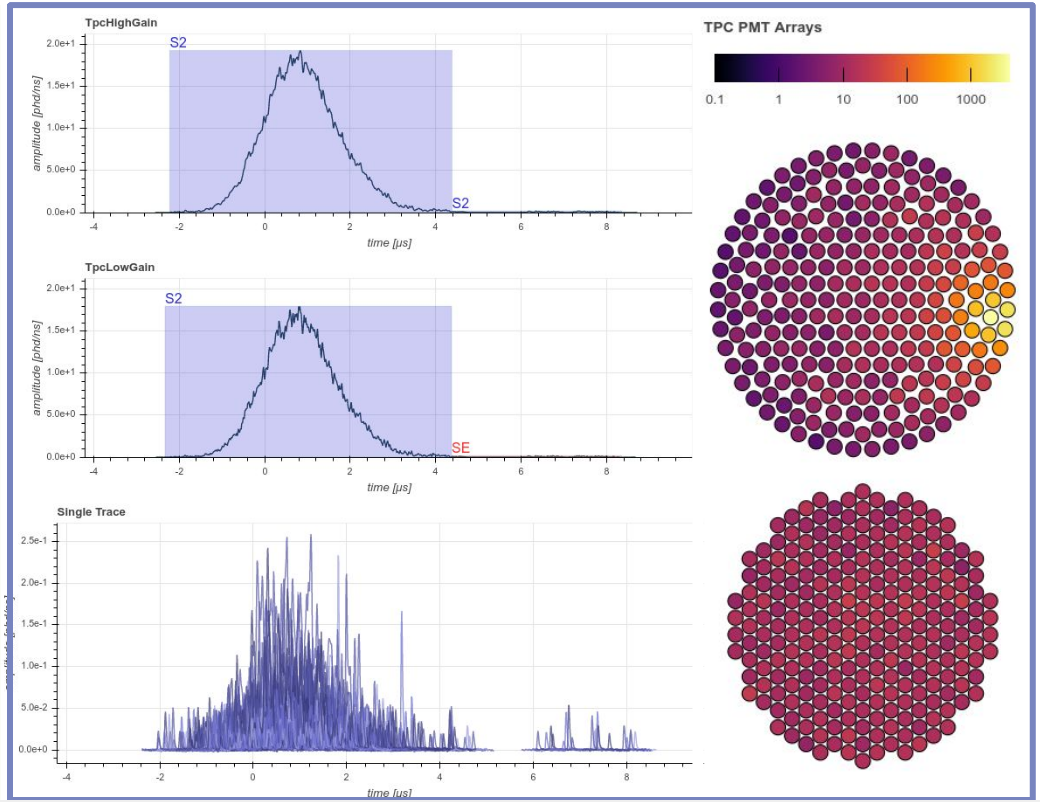


Pulses in LZ

S1 pulses

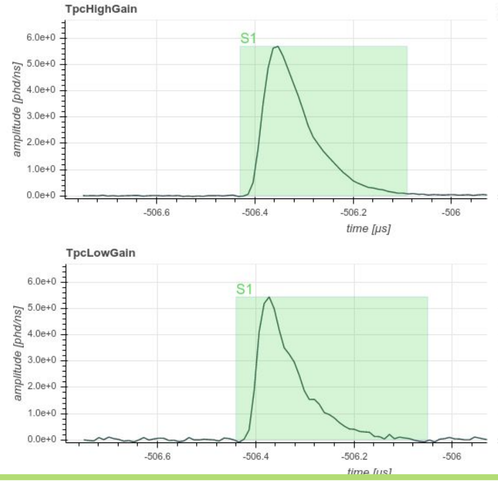


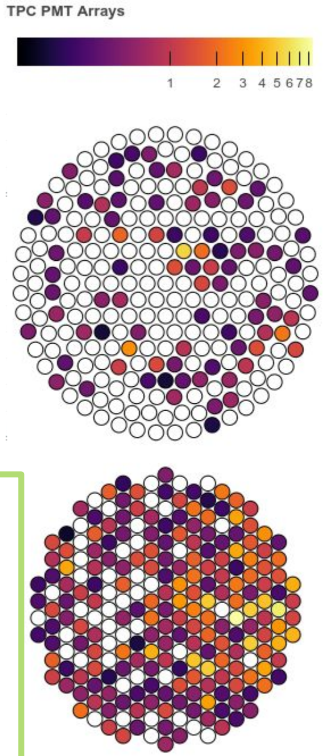
S2 pulses



Pulses in LZ

S1 pulses

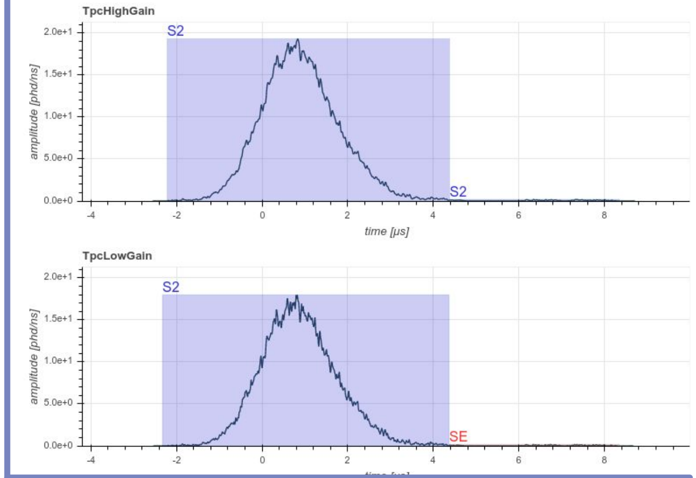


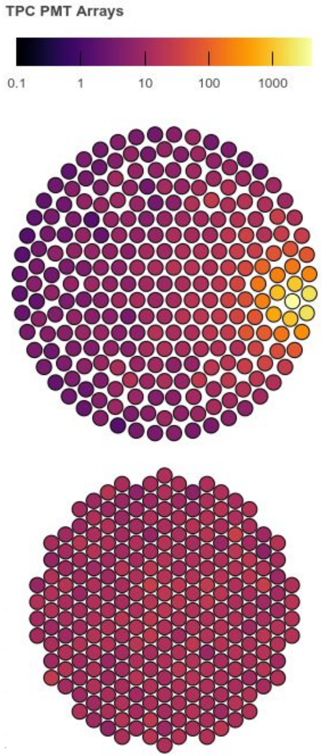


- Short (~100 ns)
- Fast rise, slower fall (exponential-like)
- Most light captured on bottom PMTs
- Typically lower area than S2s
- 3 PMT coincidence required

If coincidence=2 then its a MPE

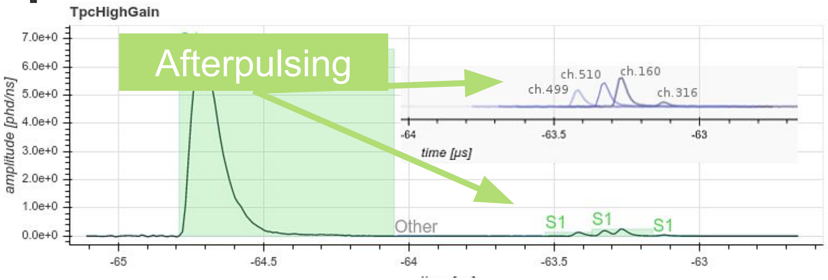
S2 pulses

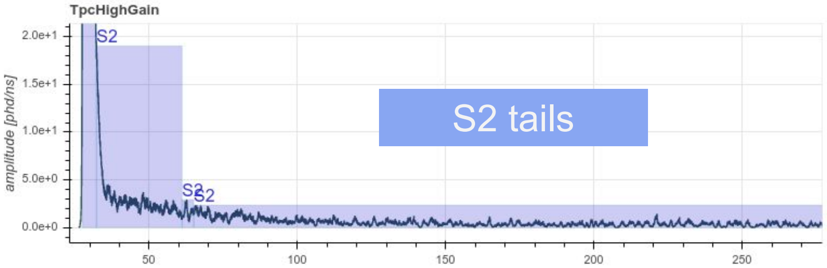


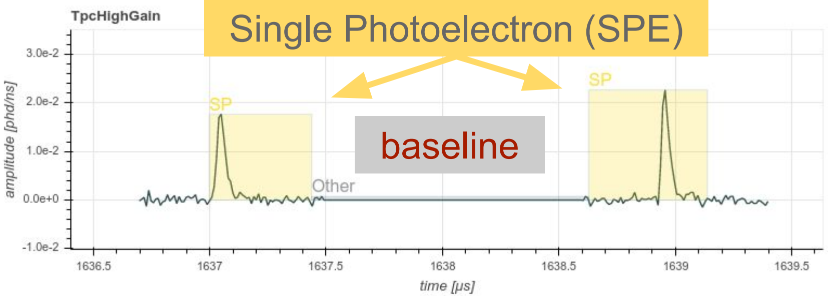


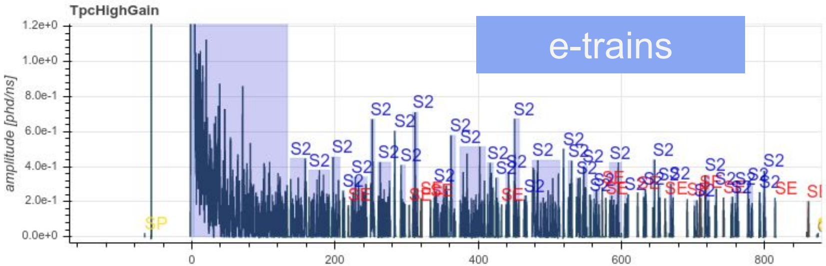
- Long (~5 µs)
- Approx. symmetric rise and fall (Gaussian-like)
- Most light captured on top PMTs
- Lower limit on area defined by **single electron** size
- Typically larger area than S1s

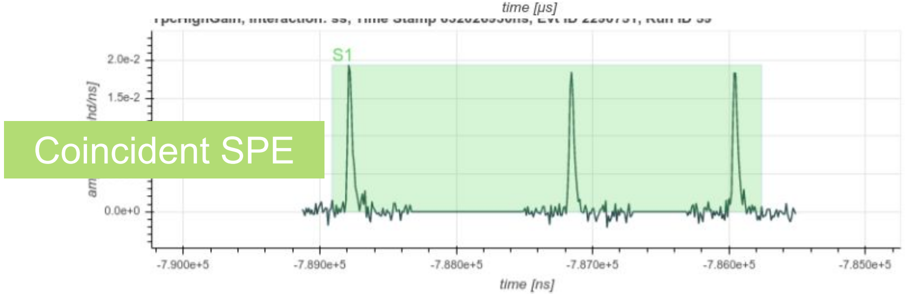
Pulses in LZ

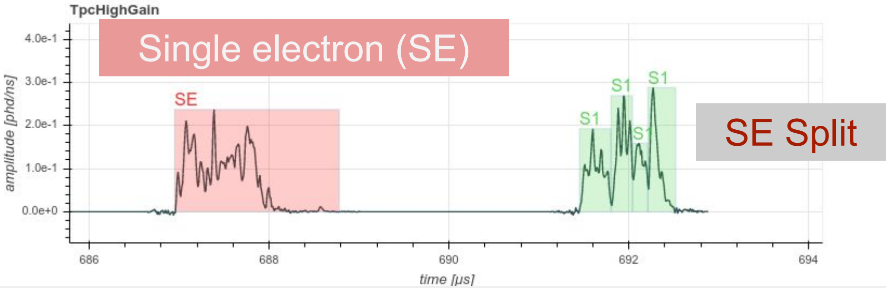

TpcHighGain
Afterpulsing
ch.510
ch.160
ch.499
ch.316
Other
S1
S1
S1
amplitude [phd/ns]
time [µs]


TpcHighGain
S2
S2 tails
S2
amplitude [phd/ns]


TpcHighGain
Single Photoelectron (SPE)
SP
SP
baseline
Other
amplitude [phd/ns]
time [µs]


TpcHighGain
e-trains
S2
SE
SP
amplitude [phd/ns]


S1
Coincident SPE
time [ns]


TpcHighGain
Single electron (SE)
SE
S1
S1
S1
S1
SE Split
amplitude [phd/ns]
time [µs]

LZap - LZ Analysis software

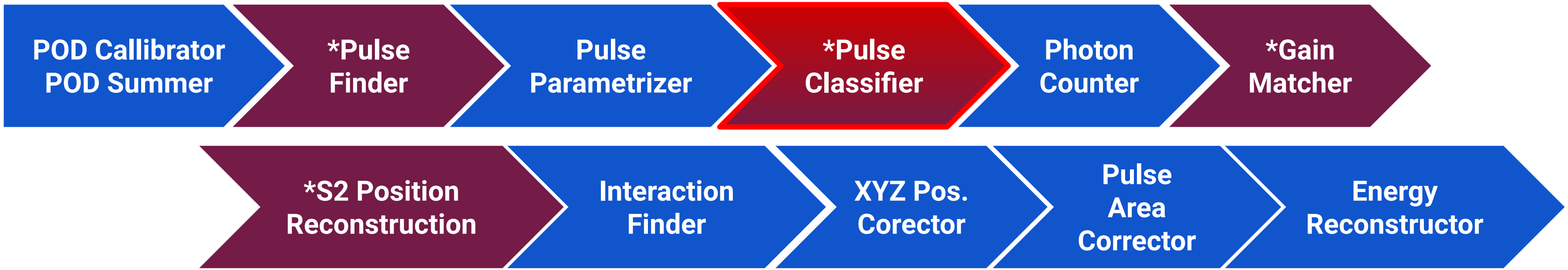
LZap Input: digitized waveforms (DAQ) - pulse only digitization (POD)

LZap Output: high-level reduced quantities (RQs)

The Pulse Classifier module is critical for all subsequent algorithms

Pulse Classifier Input: Pulse parameters from **Pulse Finder** and **Pulse Parametrizer** modules.

Pulse Classifier Output: Classifications as probability array for all classes +1 **AND** discrete class labels.



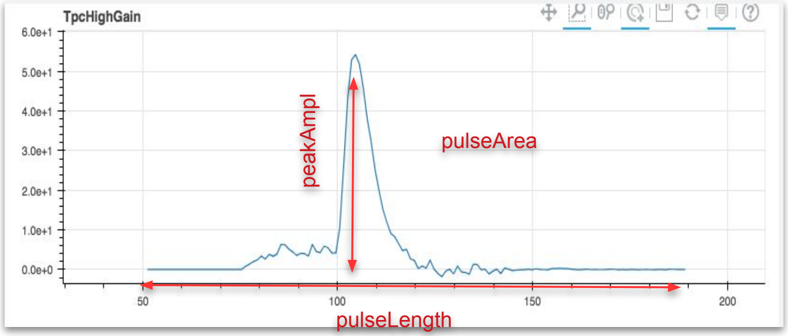
***modules developed and currently maintained by the LIP group**

Pulse classification in LZ

Pulse Classifier Input: 17 pulse parameters obtained from the *Pulse Finder* and *Pulse Parametrizer*:

- Pulse area (pA)
- Pulse amplitude (pH)
- Pulse length (pL, pL90 - length at 90% area)
- Prompt fraction (pF) fraction of area at start of pulse: 50, 100, 200, 500, 1k, 2k and 5k ns
- Top-bottom asymmetry (TBA) = $(A_{top}-A_{bot})/pA$
- Area fraction time (aft) time at X% integrated area: 5%, 25%, 50%, 75%, 95% area

Classifier Output: Probability vector for all topologies +1:
[S1, S2, SPE, SE, MPE, Other]



The pulse parameters (data features) are mostly geometrical properties of the summed waveform, obtained by the Pulse Parametrizer module.

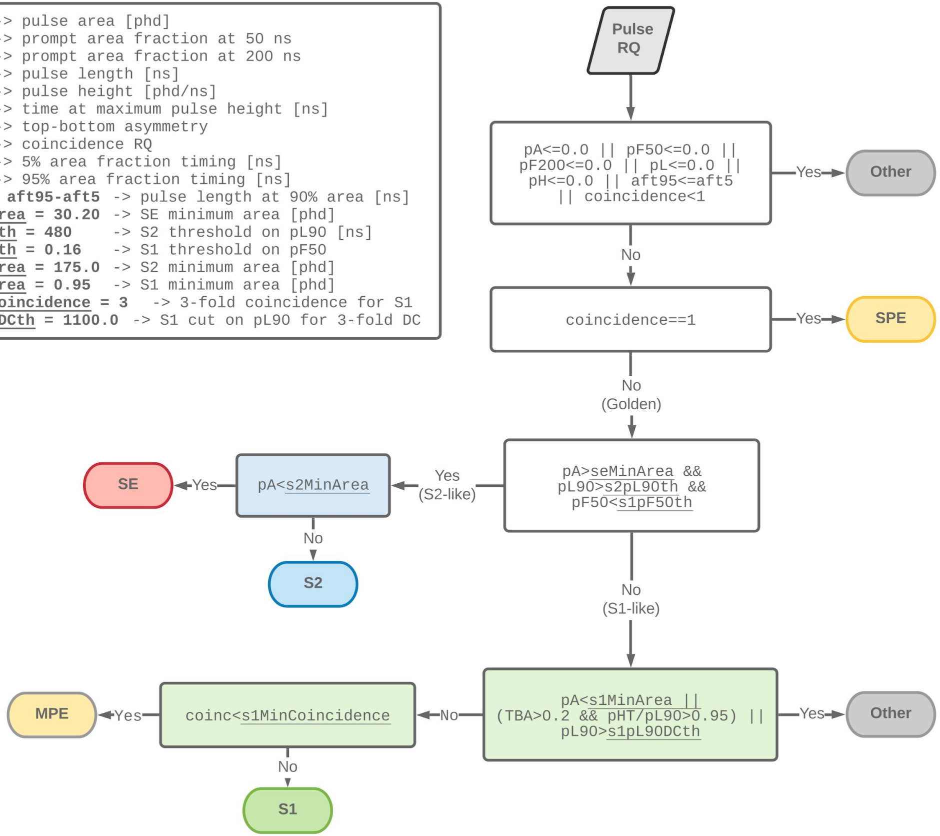
The term "parameter" here is unfortunate, these are not adjustable variables of the model!

Pulse classification in LZ

Current classifier in LZap is HADES
(Heuristic Algorithm for Discrimination of Event Substructures)

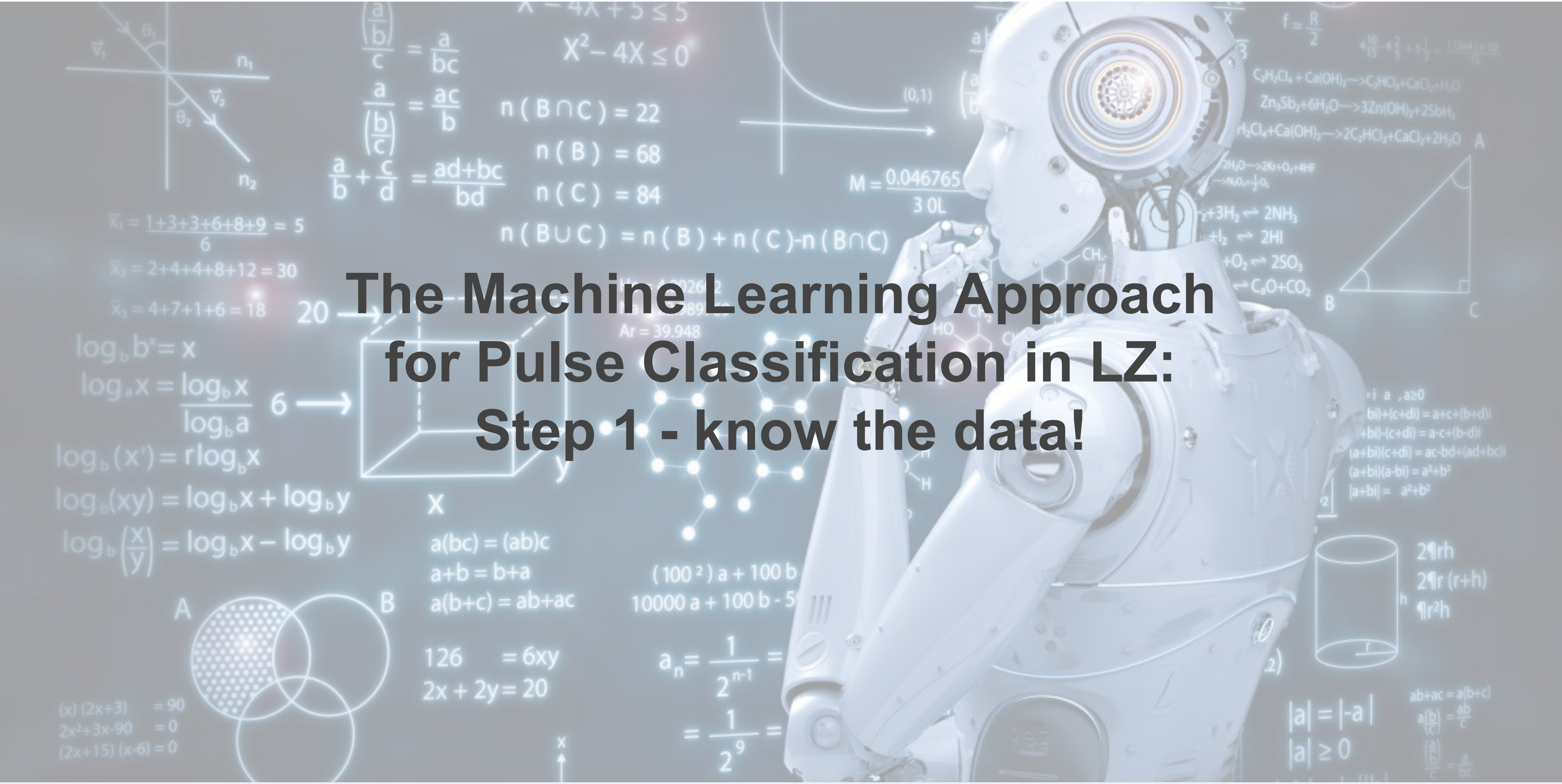
- Heuristic decision tree that is robust and easily modified
- Purely categorical and prone to bias
- Uses only 10 features. Estimated classification accuracy of **98.6%**
- Trained by hand!! More bias…

```
pA    -> pulse area [phd]
pF50  -> prompt area fraction at 50 ns
pF200 -> prompt area fraction at 200 ns
pL    -> pulse length [ns]
pH    -> pulse height [phd/ns]
pHT   -> time at maximum pulse height [ns]
TBA   -> top-bottom asymmetry
coinc -> coincidence RQ
aft5  -> 5% area fraction timing [ns]
aft95 -> 95% area fraction timing [ns]
pL90 = aft95-aft5  -> pulse length at 90% area [ns]
seMinArea = 30.20  -> SE minimum area [phd]
s2pL90th = 480     -> S2 threshold on pL90 [ns]
s1pF50th = 0.16    -> S1 threshold on pF50
s2MinArea = 175.0  -> S2 minimum area [phd]
s1MinArea = 0.95   -> S1 minimum area [phd]
s1MinCoincidence = 3   -> 3-fold coincidence for S1
s1pL90DCth = 1100.0  -> S1 cut on pL90 for 3-fold DC
```



We want to use Machine Learning to:

- Better understand the data
- Improve HADES
- Maybe develop a better classifier
    - Minimally-biased with high classification accuracy

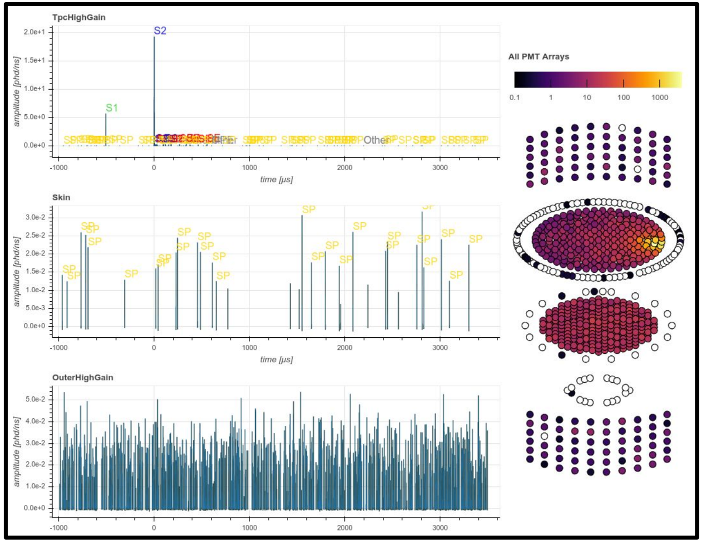


The Machine Learning Approach for Pulse Classification in LZ: Step 1 - know the data!

Step 1: examine the data

This is (arguably) the most important preprocessing step!

- Can inform on what methods may be better or may not work at all!
- May hint at future problems that some algorithms may face
- Check the natural behaviour of the data features (pulse parameters)



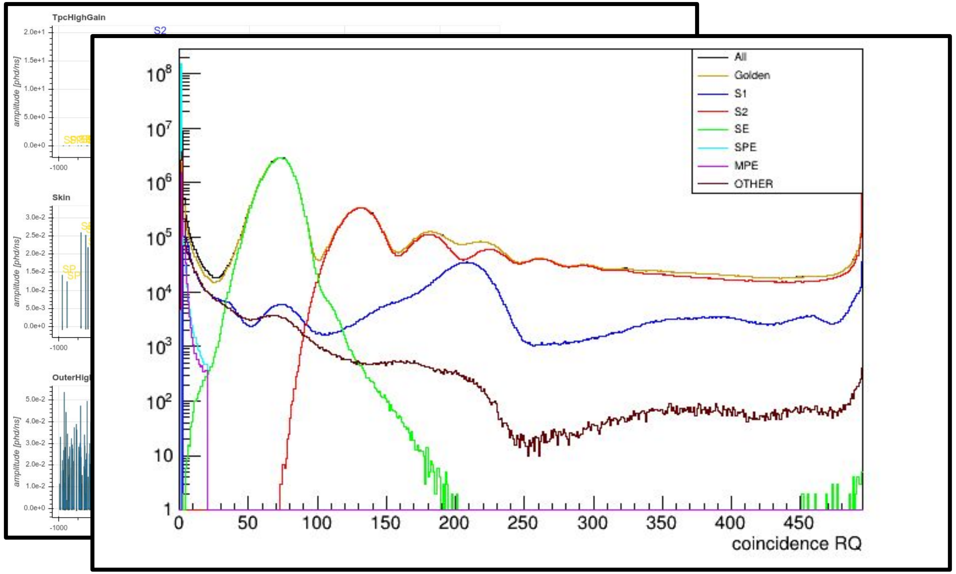
Use the tools available to explore the data objects and the features (pulse parameters)

- Event Viewer (as shown before)

Step 1: examine the data

This is (arguably) the most important preprocessing step!

- Can inform on what methods may be better or may not work at all!
- May hint at future problems that some algorithms may face
- Check the natural behaviour of the data features (pulse parameters)



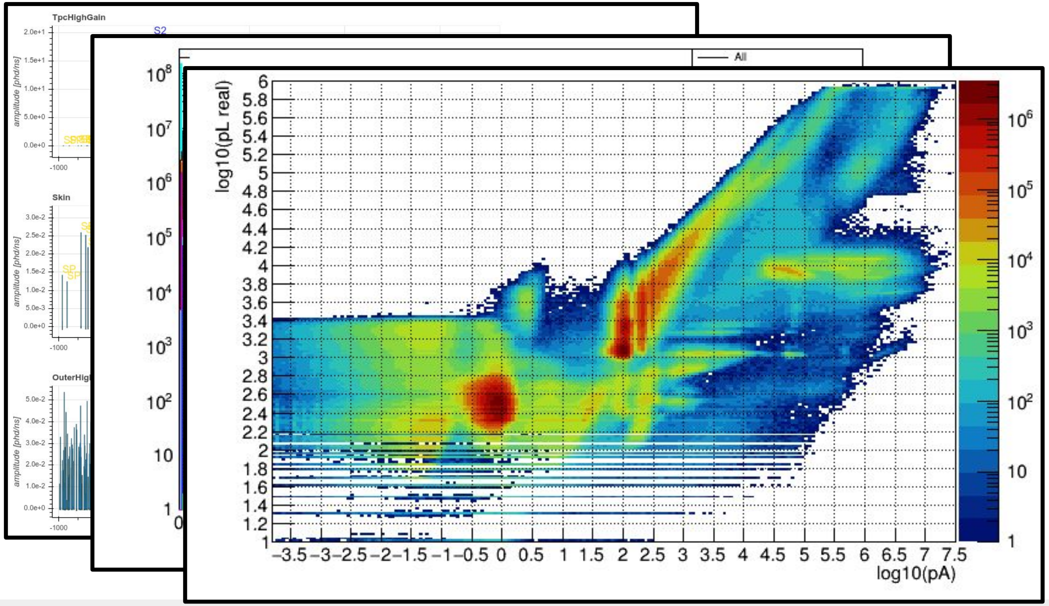
Use the tools available to explore the data objects and the features (pulse parameters)

- Event Viewer (as shown before)
- Histogram available parameters
- Identify distribution's features

Step 1: examine the data

This is (arguably) the most important preprocessing step!

- Can inform on what methods may be better or may not work at all!
- May hint at future problems that some algorithms may face
- Check the natural behaviour of the data features (pulse parameters)

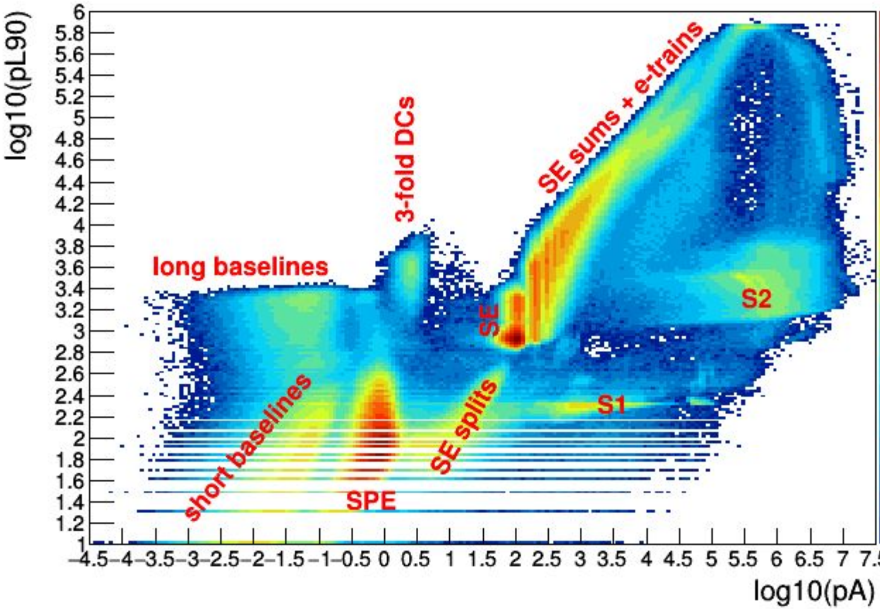


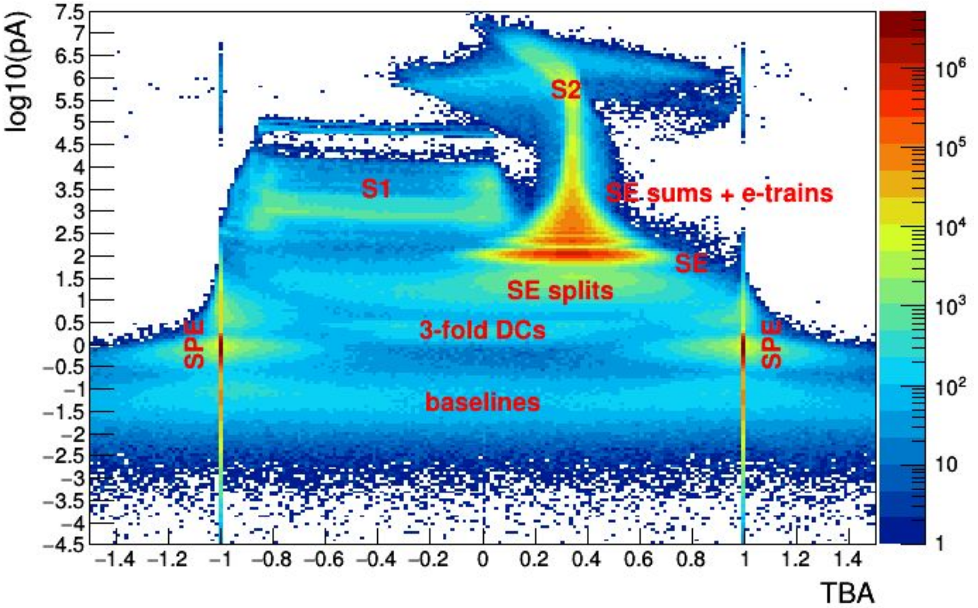
Use the tools available to explore the data objects and the features (pulse parameters)

- Event Viewer (as shown before)
- Histogram parameters
- Identify distribution's features
- Identify parameters correlations
- etc...

Step 1: examine the data

- A lot of information can be inferred just by looking at the plots;
- To be absolutely sure, select some pulses and look at the waveforms.





Step 1: examine the data

Only these matter

Class representativity:

In classification, an algorithm may devote a disproportionate amount of attention to more statistically significant populations

- Can be checked via **handscanning**: large collaborative effort involving 50+ people looking at the data
- **We can use HADES!** The classification accuracy is estimated to be at **98.6%** from handscans

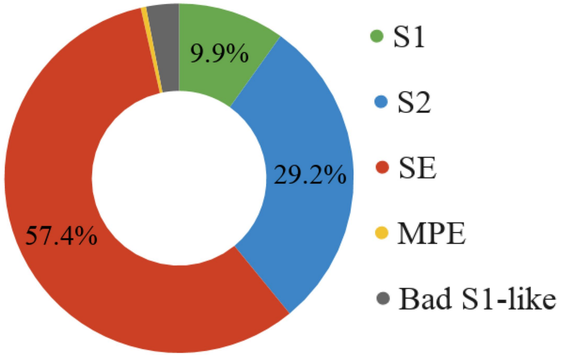
Found strong asymmetry between classes!!

- SE pulses are dominant (due to e-trains after S2s)
- S1 pulses (and MPE) are misrepresented, troubling!

Misrepresented classes can be seen as outliers and ignored.

	N pulses	event avg.	% total	% golden
All pulses	98784	131.7		
Bad pulses*	3329	4.4	3.37	
SCP	32860	43.8	33.26	
Golden	64561	86.1	65.36	
S1	6373	8.5	6.45	9.87
S2	18862	25.2	19.09	29.22
SE	37043	49.4	37.50	57.38
MPE	317	0.4	0.32	0.49
Bad S1-like*	1966	2.6	1.99	3.05
Bad S2-like*	0	0.00	0.0	0.00

*Other pulses

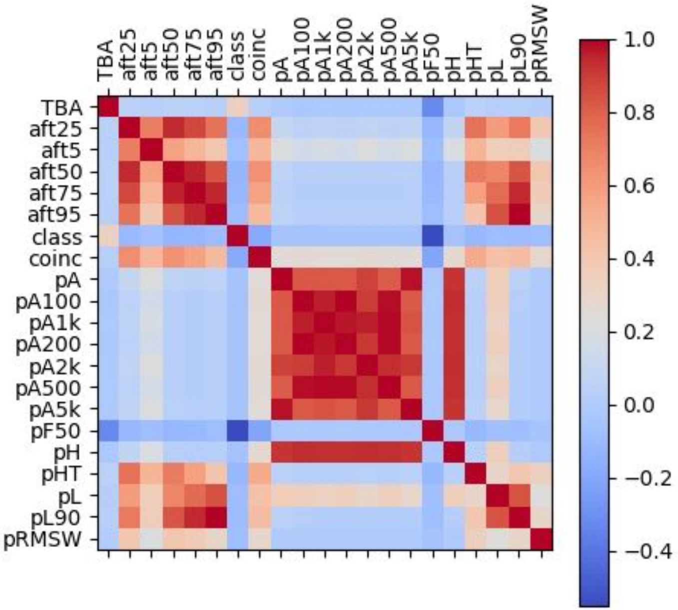


Step 1: examine the data

Pulse parameter correlation:

Using strongly correlated features results in <u>little information gain</u> when compared to the usage of only one variable

- Some parameters are highly correlated
  - Pulse areas in different time windows
  - Area fraction times and length
- Discard degenerate features?



Areas and fraction times are too degenerate!

Correlated attributes reduce discrimination power...

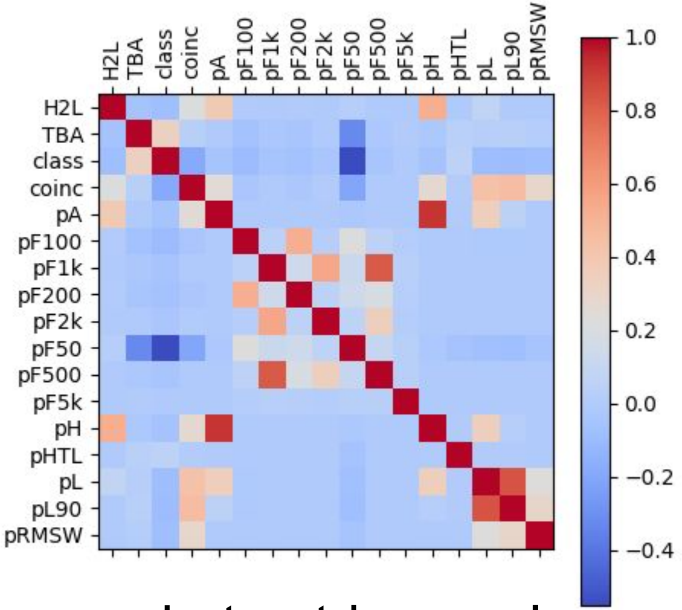
Step 1: examine the data

Pulse parameter correlation:

Using strongly correlated features results in <u>little information gain</u> when compared to the usage of only one variable

- Some parameters are highly correlated
  - Pulse areas in different time windows
  - Area fraction times and length

- Discard degenerate features?
  NO! If possible combine them in useful ways!
  - pFxx = pAxx / pA
  - pL90 = aft95 - aft5
  - H2L = pH / pL90
  - pHTL = pHT / pL90



Lost most degeneracy!

Gained discriminant power while maintaining a large parameter space.

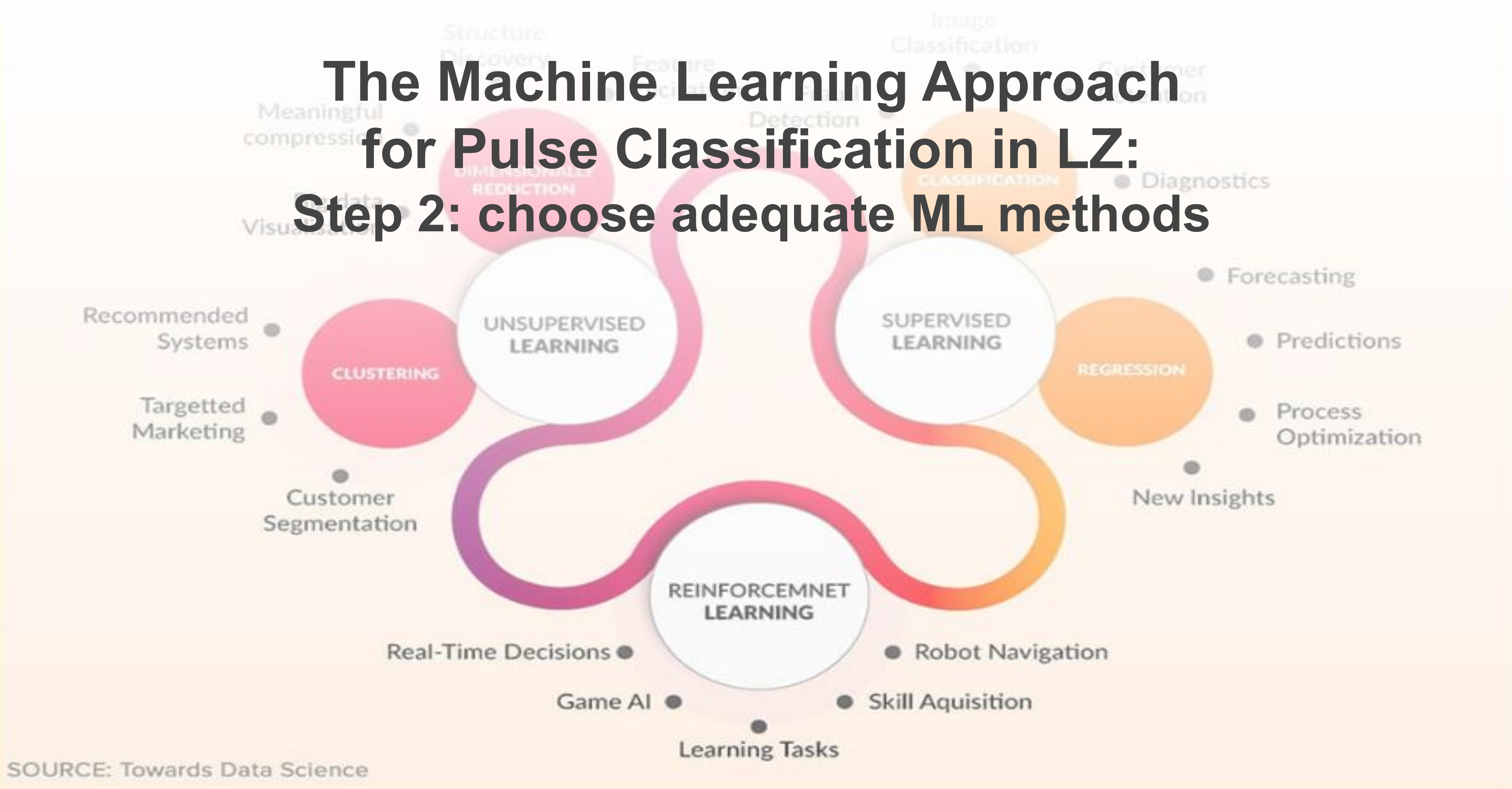
Step 1: examine the data

Is there something else we can do to skim/clean the data?

- Remove outliers not relevant for the problem?
  - data objects that are not associated with the underlying data model.
- Exclude data objects with abnormal features, i.e., data noise, accidentals?
  - miscalculations, transcription errors, poor variable precision, typos, human error, etc…
- Sample biasing?
  - sample weighing,
  - stratified sampling (sampling some parts of the data more than others).
- Curse of dimensionality?
  - Dealing with a small number of features, not really an issue here.

Stop and breathe...

The Machine Learning Approach for Pulse Classification in LZ: Step 2: choose adequate ML methods

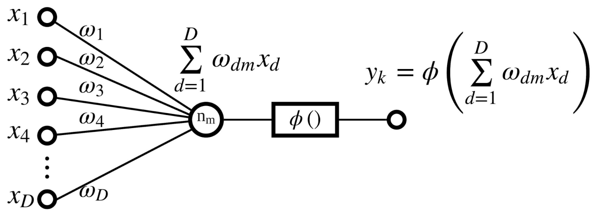


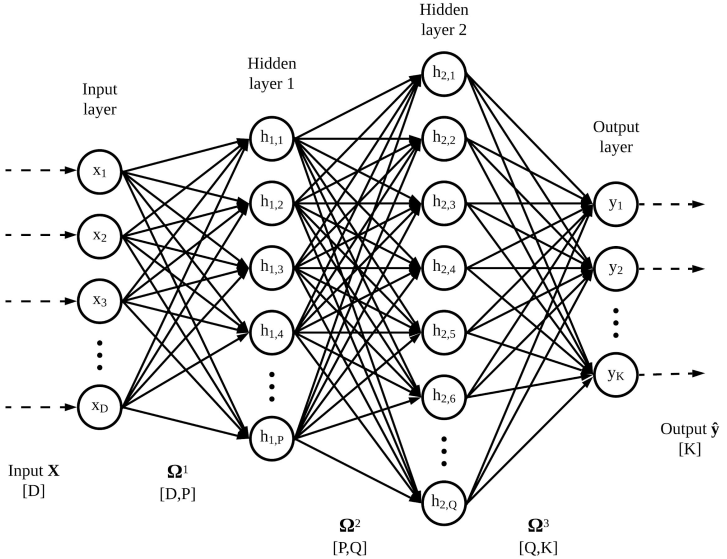
Neural Networks

Collective of interconnected units (neurons) capable of receiving, processing and communicating information with each other.

- Strength of connections are adjustable parameters (weights)
- Output of a neuron is given by the weighted sum of its inputs passed through an activation function
- A system with a relatively small number of neurons can display a large complexity
- Feed-forward networks with at least one computational layer and activation functions that are squashing functions are **universal approximators**

$$\hat{y}_n = f(\mathbf{x}_n)$$





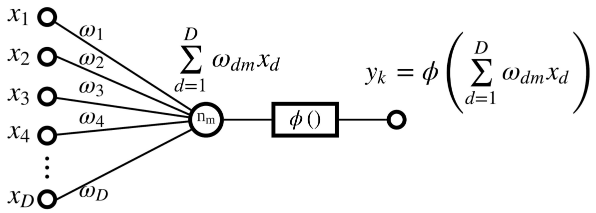
Neural Networks

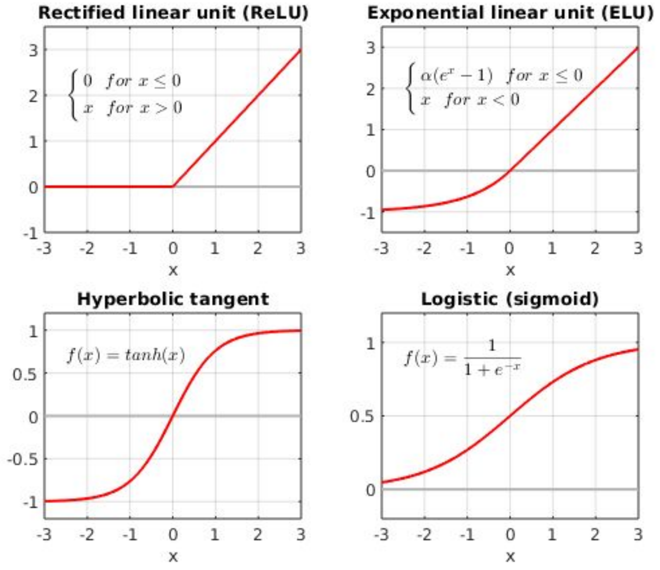
Neural Networks

Collective of interconnected units (neurons) capable of receiving, processing and communicating information with each other.

- Strength of connections are adjustable parameters (weights)
- Output of a neuron is given by the weighted sum of its inputs passed through an activation function
- A system with a relatively small number of neurons can display a large complexity
- Feed-forward networks with at least one computational layer and activation functions that are squashing functions are **universal approximators**

$$\hat{y}_n = f(\mathbf{x}_n)$$

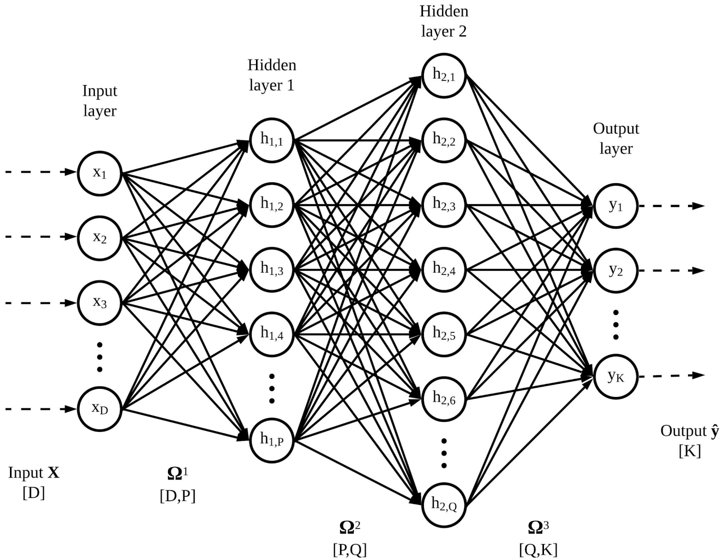




Neural Networks

Stochastic gradient descent with backpropagation

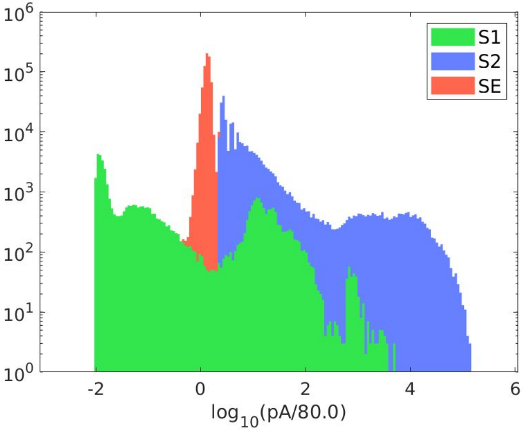
1. **Forward pass:** From a randomly selected batch of labelled training data, compute $\hat{y}_n = f(\mathbf{x}_n)$ with the current weights
   a. Calculate the total loss $L(y_n; \hat{y}_n)$
   b. Calculate loss δ_L for the final layer, L
2. **Backward pass:** For each hidden layer, starting from the last
   a. Compute the loss at current layer δ_{L-1} using the loss from the previous (following) layer δ_L
   b. Compute the gradient at current layer
3. Update the weights, moving down the gradient an certain amount defined by a <u>learning rate</u> α
4. Repeat until loss converges or rebounds (early stopping)



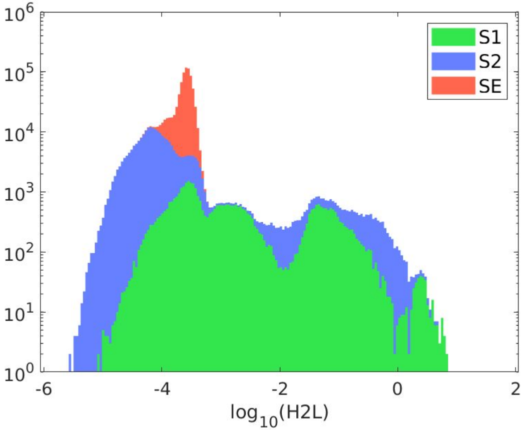
Data preprocessing:

It is important to determine if the input data is in a state that is apt to be handled by the model before training begins. Usually we just <u>mean-center</u> and <u>normalize</u>

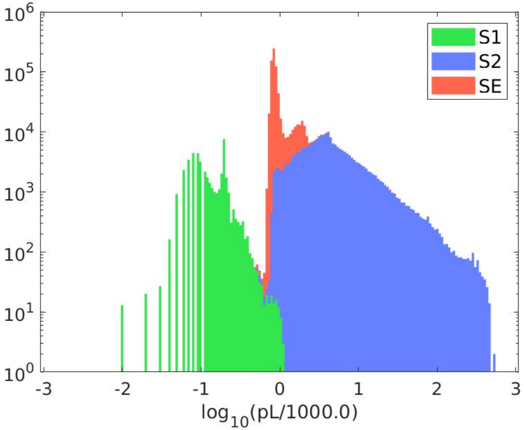
Pulse Area log centered to SE size



Height-to-Length log



Pulse Length 90% log centered to SE length



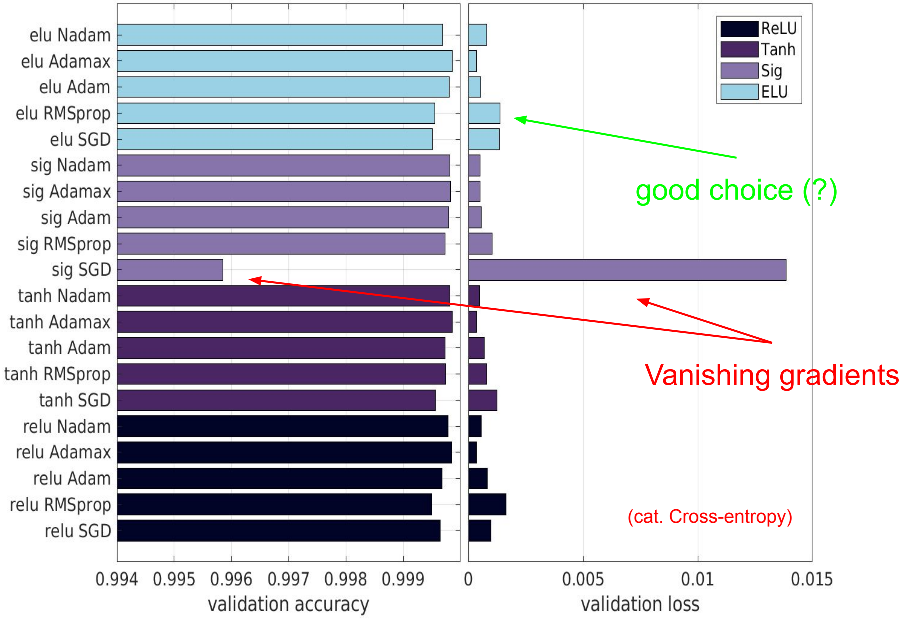
$$\mathbf{x} = \{\, \mathbf{log_{10}(pA/80.0)} \;;\; \mathrm{pF50} \;;\; \mathrm{pF100} \;;\; \mathrm{pF200} \;;\; \mathrm{pF1k} \;;\; \mathrm{TBA} \;;\; \mathbf{log_{10}(pL90/1000.0)} \;;\; \mathbf{log_{10}(pH)} \;;\; \mathbf{log_{10}(H2L)} \,\}$$

Neural Networks

Tuning of relevant hyperparameters:

- Learning rate
  - how big are the updates to the weights
- Batch size
  - how many samples are used to update the model once
- Epochs (training iterations)
  - The number of cycles through all training data
- Optimizers
  - Control gradient descent
- Number of hidden layers
- Number of hidden units per layer
- Unit non-linearity
  - activation functions

Activation function vs Optimizer



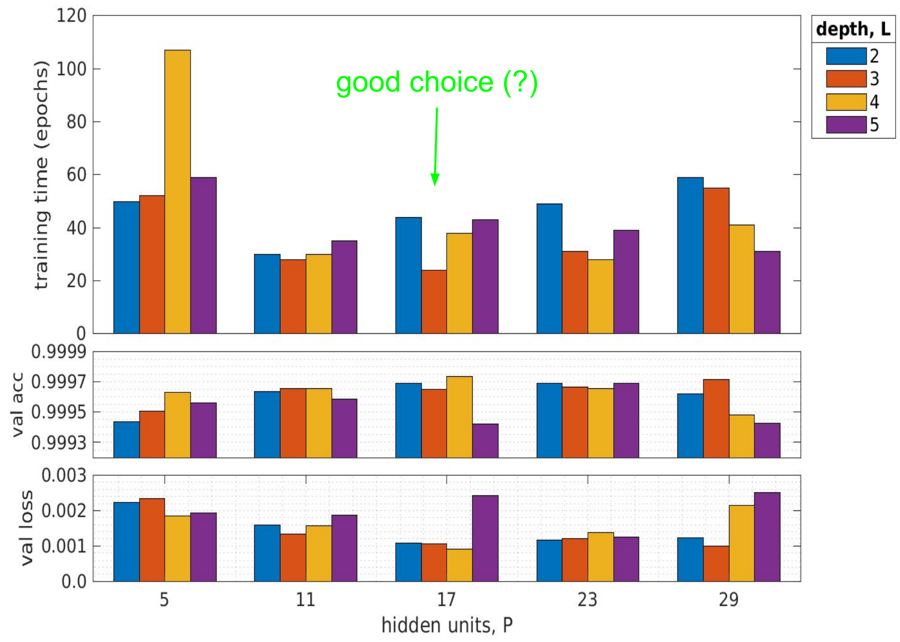
No clear difference (aside from sigmoid+SGD)

Neural Networks

Tuning of relevant hyperparameters:

- Learning rate
  - how big are the updates to the weights
- Batch size
  - how many samples are used to update the model once
- Epochs (training iterations)
  - The number of cycles through all training data
- Optimizers
  - Control gradient descent
- Number of hidden layers
- Number of hidden units per layer
- Unit non-linearity
  - activation functions

Size of the NN



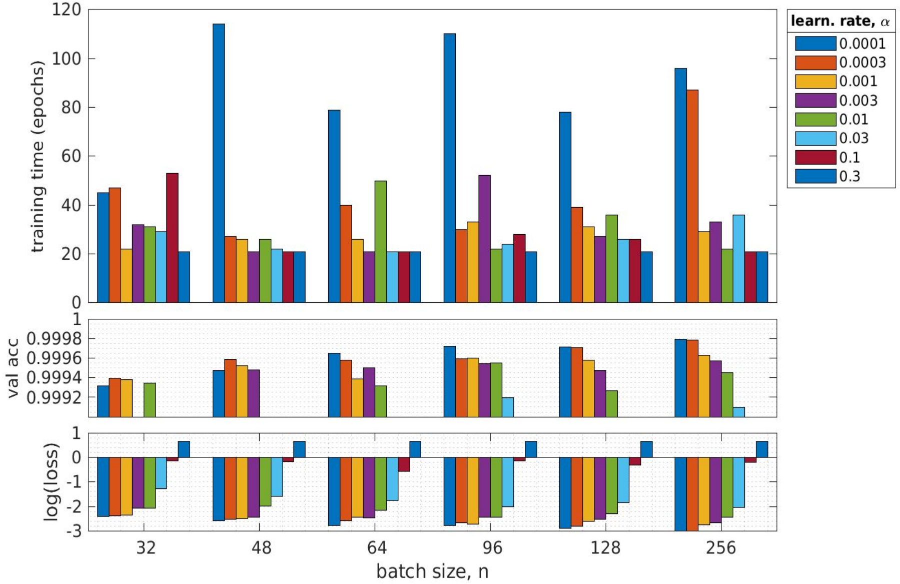
Again, no clear difference

Neural Networks

Tuning of relevant hyperparameters:

- Learning rate
  - how big are the updates to the weights
- Batch size
  - how many samples are used to update the model once
- Epochs (training iterations)
  - The number of cycles through all training data
- Optimizers
  - Control gradient descent
- Number of hidden layers
- Number of hidden units per layer
- Unit non-linearity
  - activation functions

Batch size vs learning rate



Large α leads to instability, small α to long training times

Good choice would be α = 0.001 and n = 128 (?)

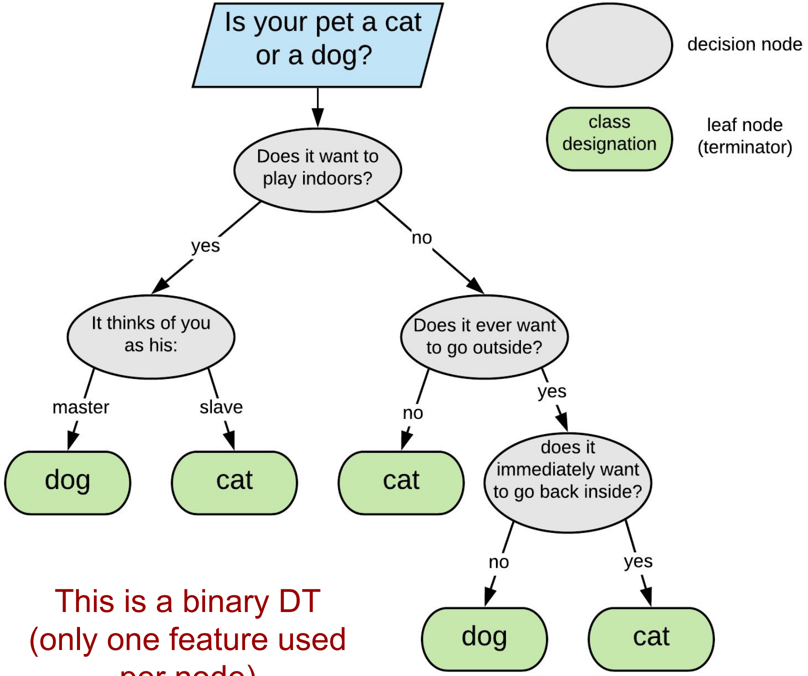
Ensemble Decision Trees

Decision trees

Flowchart-like structures used for decision making, categorization and regression analysis.

- Simple set of rules control the flow of data
- Arrange data objects in discrete categories
- **Recursive partitioning**
    - minimizing an impurity function

A subset of data objects that end in the same termination of the tree must have a similar set of properties



This is a binary DT (only one feature used per node)

Ensemble Decision Trees

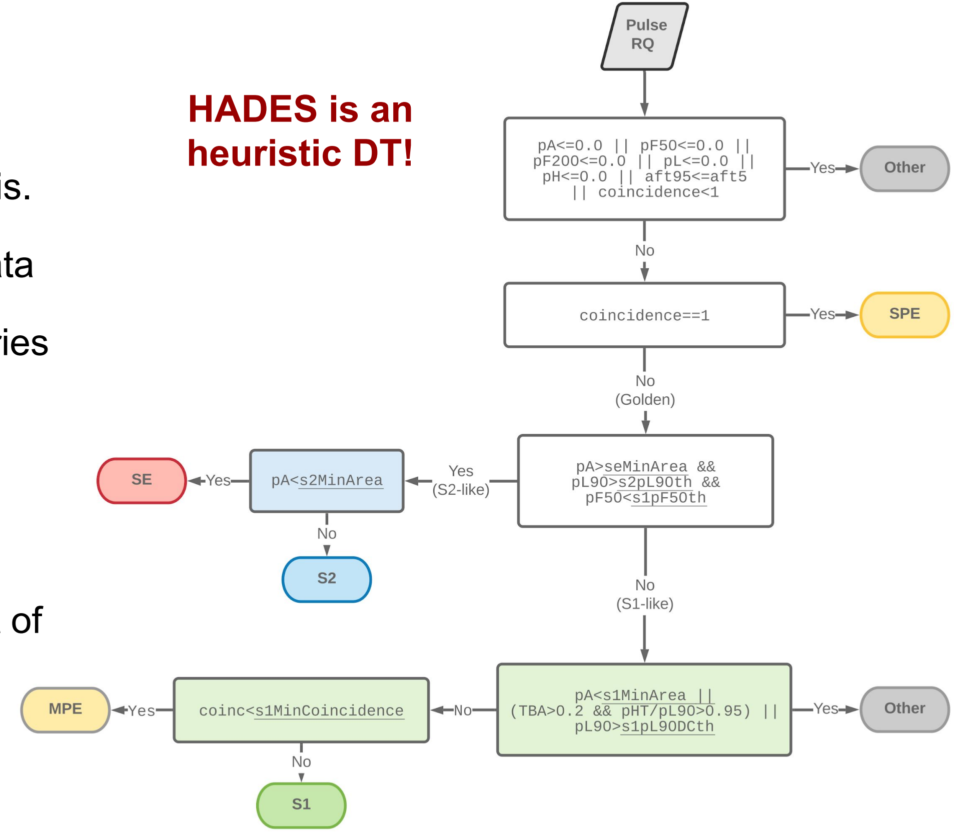
Decision trees

Flowchart-like structures used for decision making, categorization and regression analysis.

- Simple set of rules control the flow of data
- Arrange data objects in discrete categories
- **Recursive partitioning**
  - minimizing an impurity function

A subset of data objects that end in the same termination of the tree must have a similar set of properties

HADES is an heuristic DT!



Ensemble Decision Trees

Decision trees

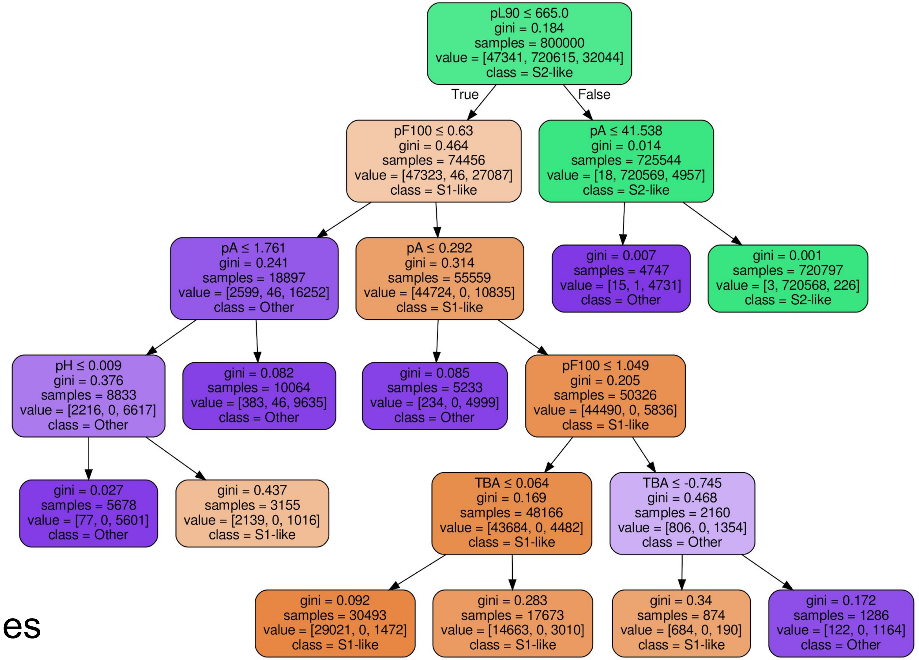
Impurity measure: GINI index

$$\sum_{k \neq k'} \hat{p}_{mk}\hat{p}_{mk'} = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}).$$

- Differentiable
- Beware of misrepresentation!

Growth and pruning:

- A tree can be grown until all nodes are pure nodes
  - Not desirable, probably overfitting
- Limit depth, leaf samples and split samples
- Pruning: removal of internal decision nodes with low separation power



Ensemble Decision Trees

Ensemble of trees (forest)

Random Forests (RFs) are overfitting-resistant!

- Trained using weak learnability
- Often use majority vote as global output
- Bootstrap aggregation → **Random Forests**
    - Each tree is trained with a random subset of data
    - **Feature bagging -** sample the features each tree can use as well!
    - Out-of-Bag (OOB) sampling
- Boosting → **Boosted DTs**
    - Sequence of weak learners trained with data weighed with the errors of the previous learner

Ensemble methods can be used to estimate **feature importance**

Interesting way of finding the best discriminant features

Ensemble Decision Trees

The Random Forest Classifier

Input conditioning:

$$\mathbf{x} = \{\, \mathbf{pA} \,;\, \mathbf{pF50} \,;\, \mathbf{pF100} \,;\, \mathbf{pF200} \,;\, \mathbf{pF1k} \,;\, \mathbf{TBA} \,;\, \mathbf{pL90} \,;\, \mathbf{pH} \,;\, \mathbf{pHTL} \,;\, \mathbf{pRMSW} \,\}$$

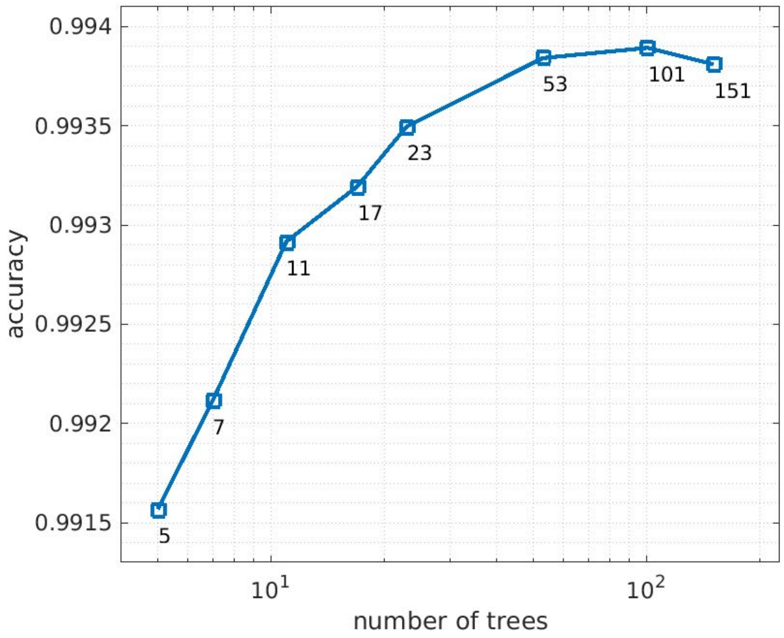
- No need to modify the parameters since binary DTs will use simple thresholds
- Extended parameter space in order to study **feature importance**
- High-dimensional data with **nuisance features will hinder performance**
- Careful with representativity when bagging!
    - Misrepresented classes can be suppressed in bootstrapped sets

Ensemble Decision Trees

The Random Forest Classifier: Tuning the model

Size of the forest:

- Considered fully grown trees with as little as 2 samples per decision node
- Small trees already have decent accuracy
- Little accuracy improvement above ~50 trees
    - Chosen nTrees = 101

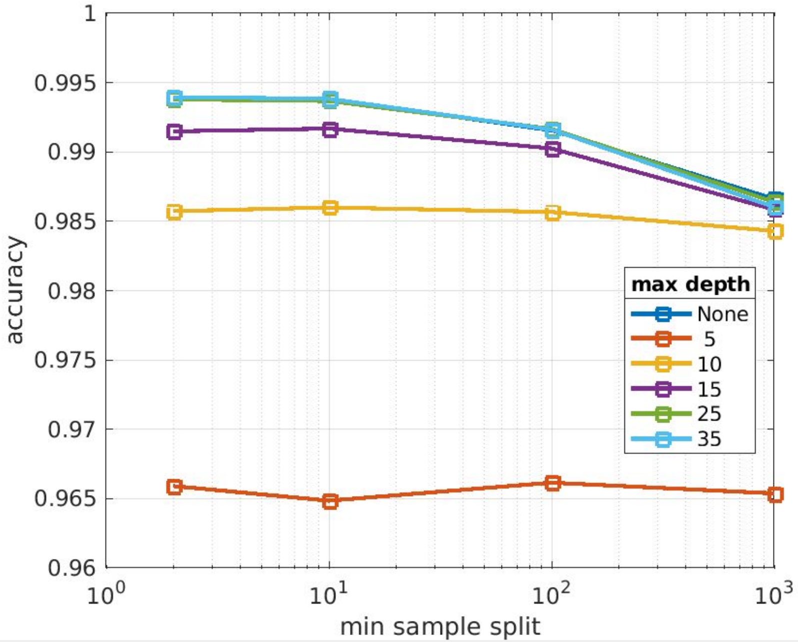


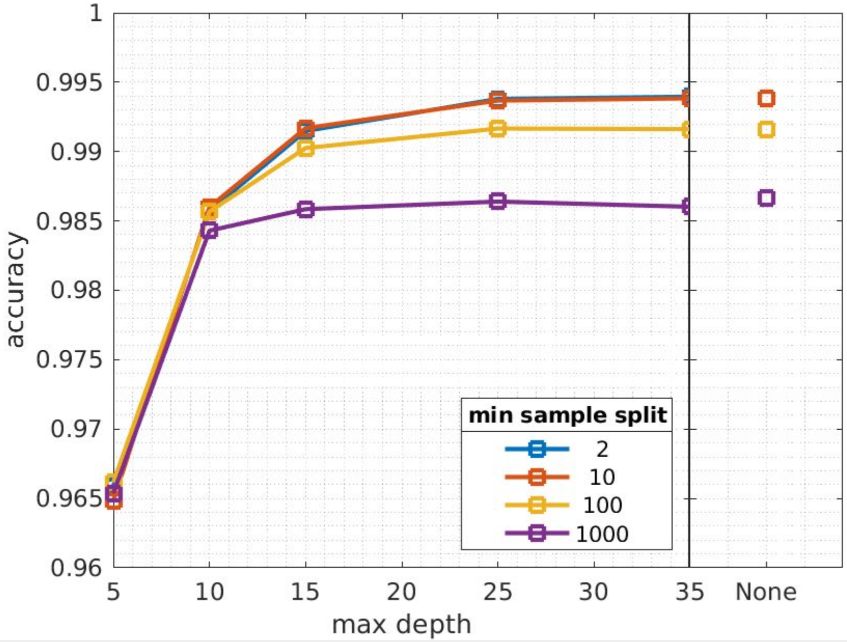
Ensemble Decision Trees

The Random Forest Classifier: Tuning the model

Depth and sample split

Chose maxDepth=None and minSamplesSplit=2





Ensemble Decision Trees

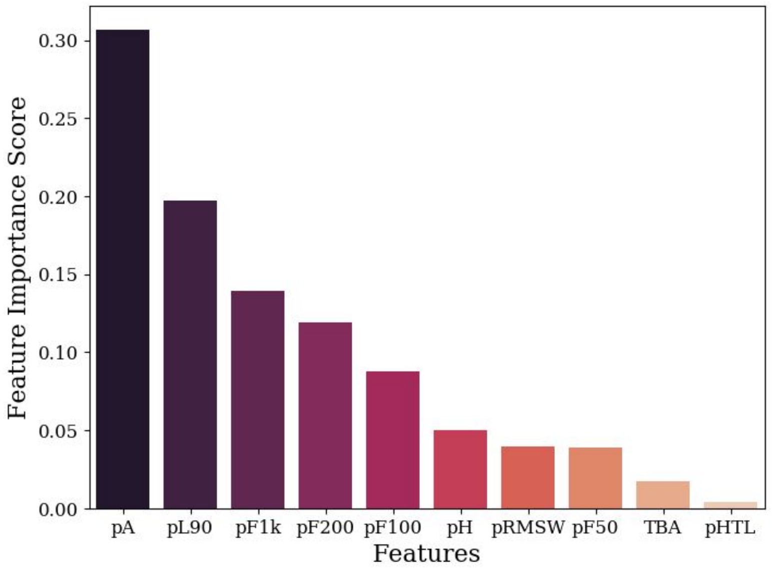
The Random Forest Classifier

Feature Importance:

- Area seems to be the overall strongest discriminant, followed by length
- top-bottom asymmetry seems to be less important

Are we happy? Is this what we want? **NO!**

- Feature importance is extremely sensitive to asymmetric representation of class labels, especially in multi-class problems
- SE abundance highly inflates pulse area importance
- **Also strongly dependent on correlations**

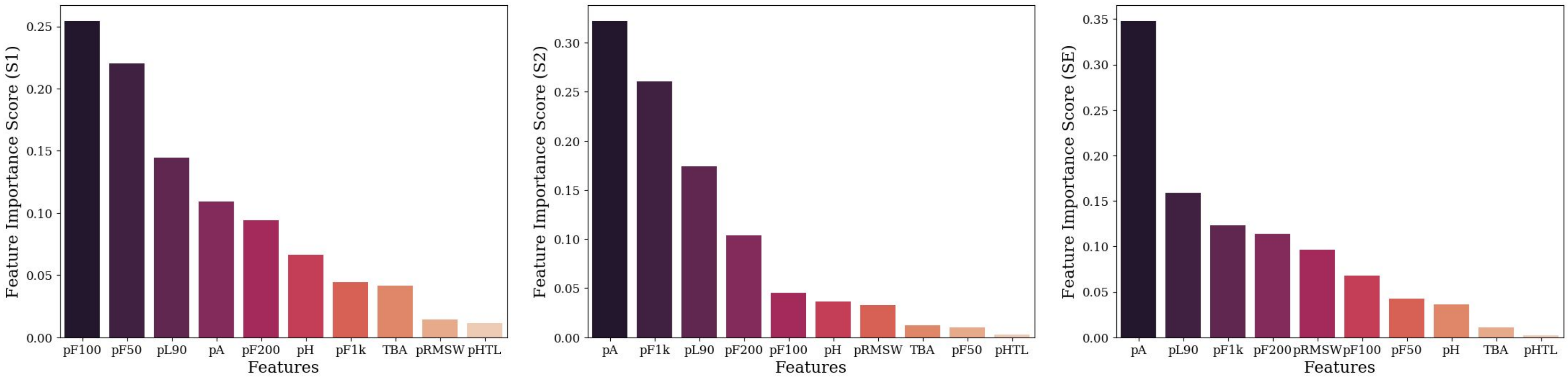


Ensemble Decision Trees

The Random Forest Classifier

Feature Importance: checking One-vs-All importance (binary classification)

- SE and S2 discriminants are similar, but S1 discriminants are very different!
- Clearly looking at overall importance is not useful...



Ensemble Decision Trees

The Random Forest Classifier

Permutation Importance: randomly permuting variables in a tree, which is guaranteed to reduce the efficiency, and compare the resulting accuracy with the one from the intact tree.

- Yields a misclassification rate that can be interpreted as the overall effect of the variable on the accuracy, and thus its importance.
- Can also be performed by noising the variables.
- Accounts for highly correlated variables in the data
- **Superior to the previous feature importance score**

Ensemble Decision Trees

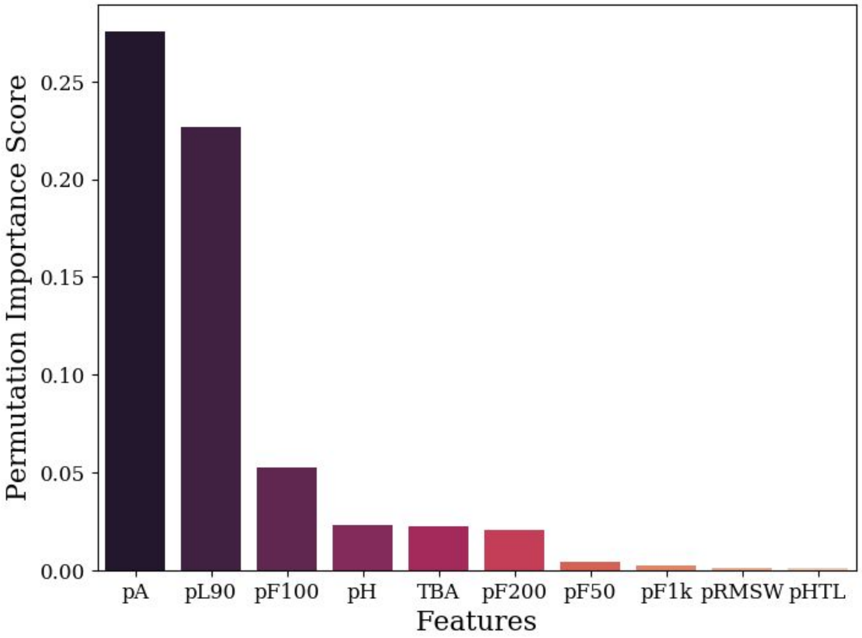
The Random Forest Classifier

Permutation Importance:

- Pulse area and length are the most important discriminants, by far
- Clear preference for pF100 over other prompt fractions
- Pulse height and TBA are ranked a bit higher now

This is a more satisfying result.

But… All parameters can serve a purpose in this analysis!



Credits

- P. Braz, “Sensitivity to the 0νββ decay of ^{136}Xe and development of Machine Learning tools for pulse classification for the LUX-ZEPLIN experiment”, PhD thesis 2020.
- P. Braz, “Machine Learning tools for pulse classification in LZ”, Ciência dos Dados em Física, 2021.

Hands On tutorial

1. (if you don’t have one already) create a Google account;
2. Download into your computer the jupyter notebook:
    a. DataScience_tutorial.ipynb
3. At your Google Drive create a directory (e.g. DataScience) and save inside:
    a. Data_Challenge_1.csv
4. Go to http://colab.research.google.com (register if necessary);
    a. Open the ***DataScience_tutorial.ipynb*** notebook saved previously;
5. … start analysing!
    a. Instruction on how to access/use ***Data_Challenge_1.csv*** are supplied in the notebook.

Challenge

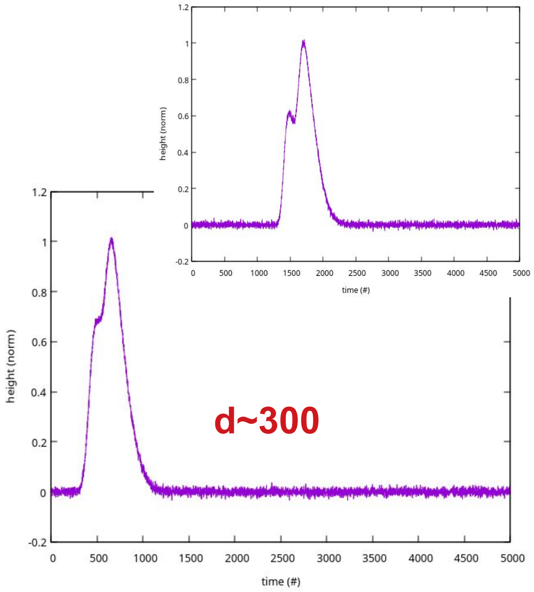
- Previously, in order to identity the type/class of the pulses registered in LZ using Machine Learning (ML), we recur to "*classical*" pre-processor modules to, for instance:
    - Identified the pulse boundaries (i.e. start and end times) in the timeline;
    - Parameterize each pulse (e.g. area, height);

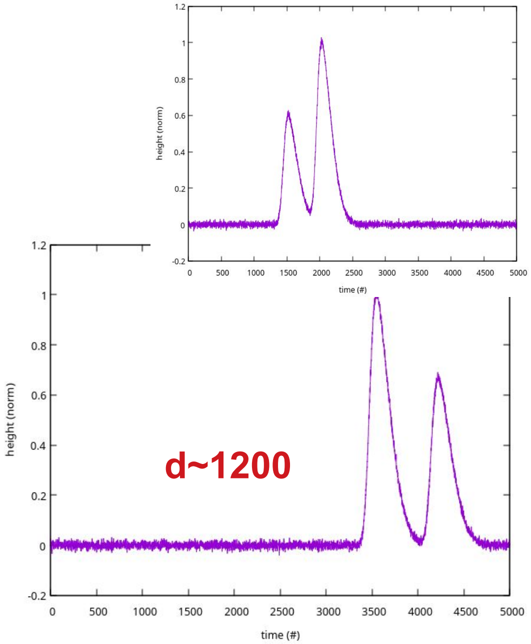
- But … would it be feasible to **extract relevant information** from the timeline directly also using ML and go without all explicit pre-processing? That's what we propose **you to try** with the following **Challenge**!
    - Also, in order to try something different and **explore new tools**, we will move from a **categorical** to a **regression** problem/analysis.

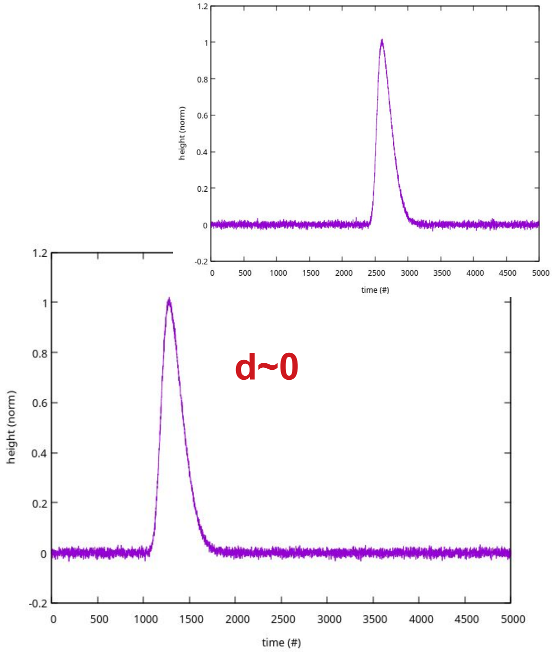
- Last, but not least: the LZ detector will also be used for other rare event searches besides Dark Matter (or WIMPs). So we also propose to explore something which is relevant for the **neutrinoless double beta decay (0υ2B) searches in LZ** , i.e.:
    - distinguish events corresponding to the emission of 1e^- from 2e^-in the decay of ^{136}Xe (without neutrino).

Challenge

- A set of synthetic waveforms were generated having 2 S2-lile pulses. For each timeline, the **start** time of the 1st pulse, **distance** (d) between the 2 pulses, relative **height** and **widths** were varied randomly within realistic values.







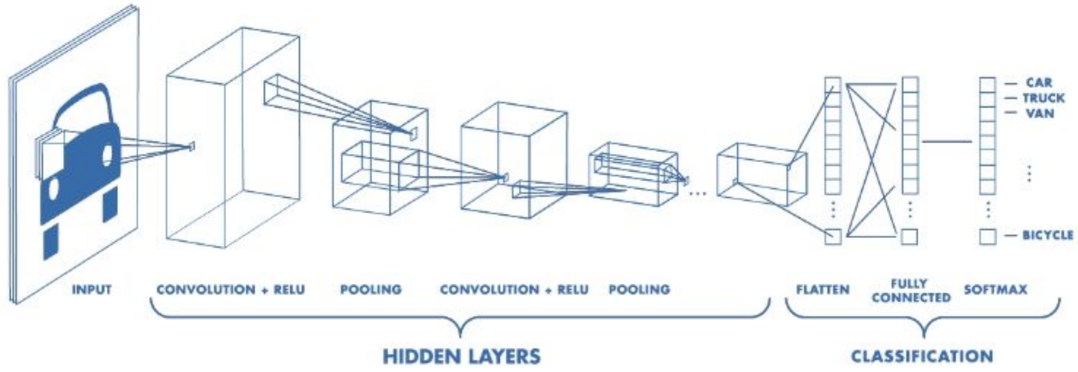
Challenge

1. At your Google Drive create/reuse a directory (e.g. DataScience) and save inside:
   a. input_2x_S2.txt : waveforms
   b. output_2x_S2.txt : distance correspondent to a. (d)
   c. input_1x_S2.txt : waveforms
   d. Output_1x_s2.txt : distance correspondent to c. (d=0)
2. Sug: train a CNN to learn/estimate the distance between 2 pulses:
3. A second set of waveforms with only 1 pulse (c. and d.) was also generated.
   a. Sug: Histogram the response of your model,
   b. Interpret the results.

Note: the use of a CNN model is suggested but not mandatory. Students are free to explore other methods to accomplish the same task. We can later compare the pros/cons of the difference approaches.

Challenge: CNN (extra)

2D Convolutional Neural Network (CNN) architecture



https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html

In our data, 2D data/filters are replaced by 1D (timeline)