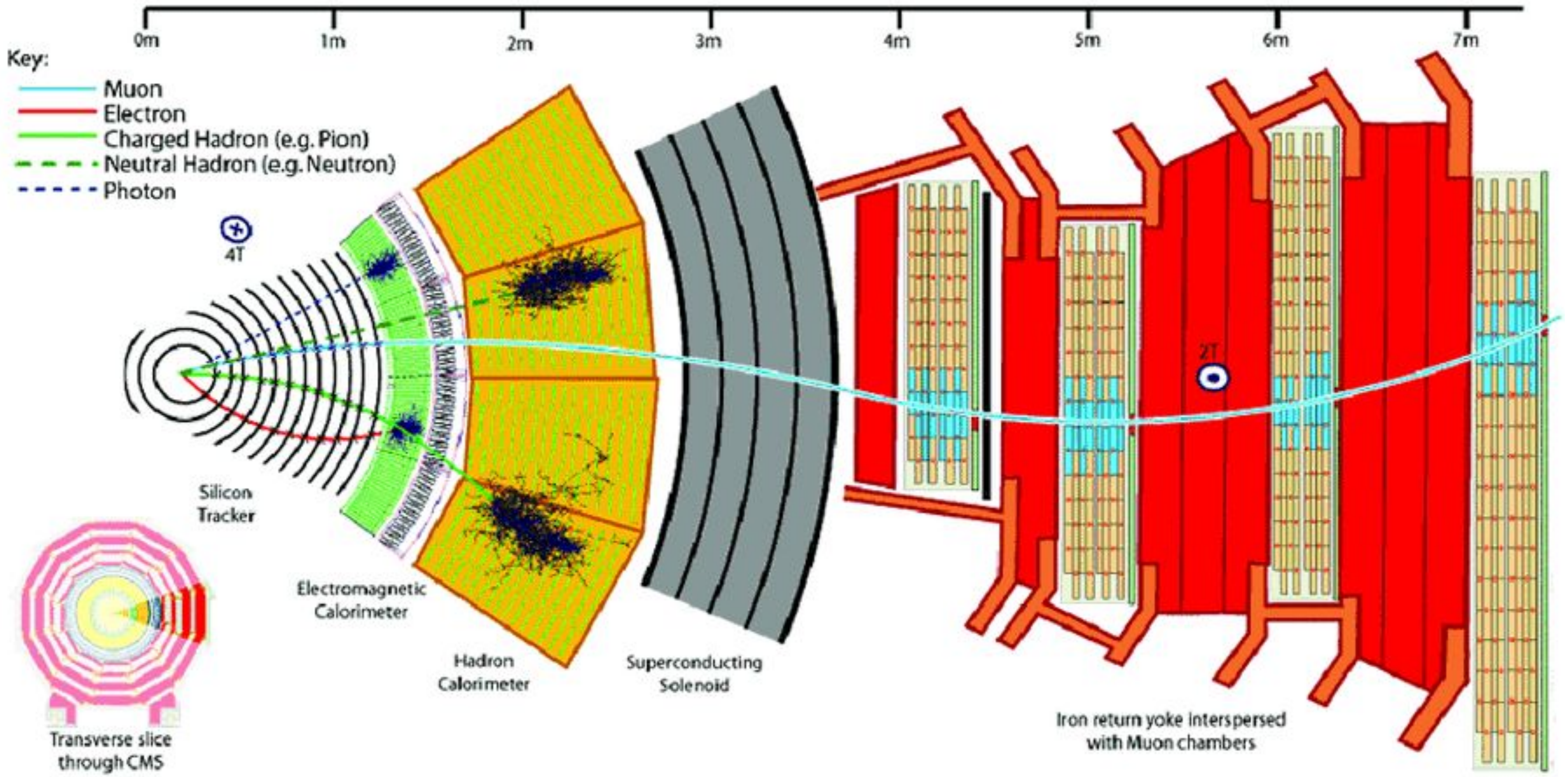
Lectures on common analysis techniques in flavour physics

IPHC, Strasbourg - 08/11/2023

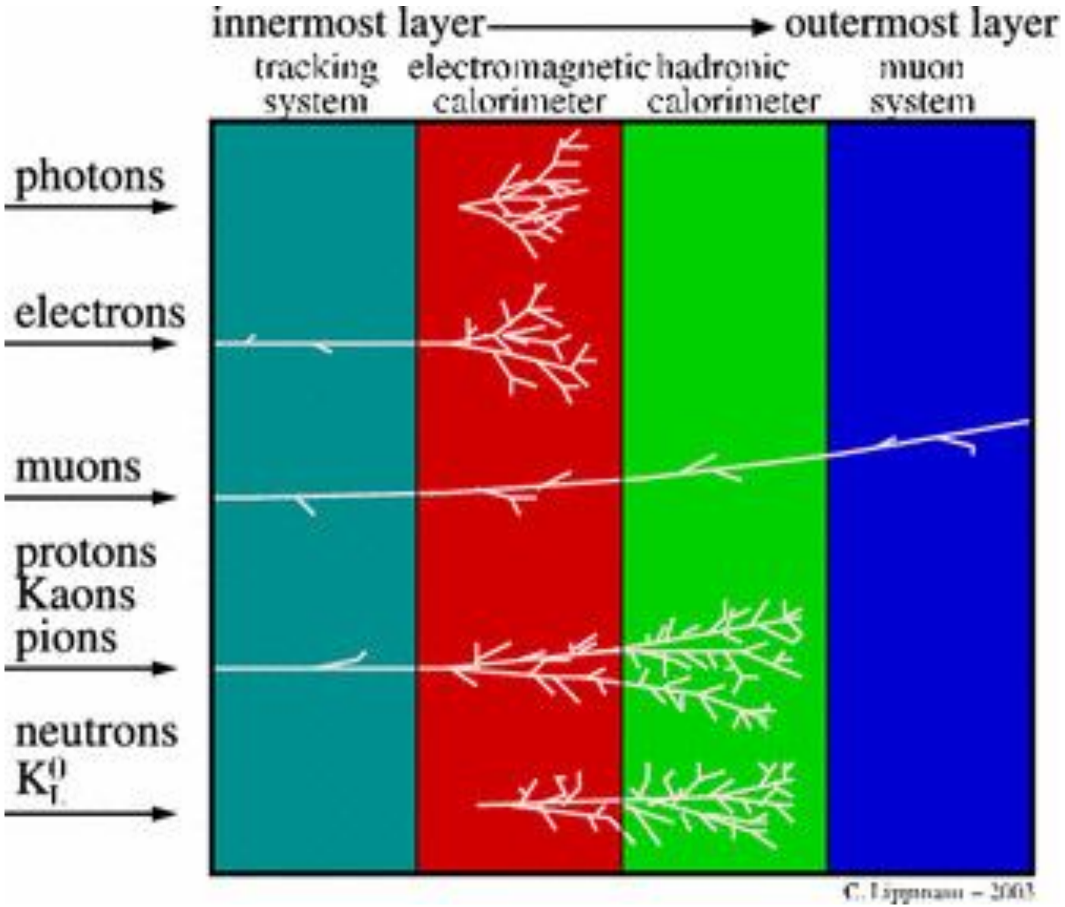
Event selection and multivariate classification

Lorenzo Capriotti
Università di Ferrara and INFN Ferrara

Particle detectors


0m
1m
2m
3m
4m
5m
6m
7m
Key:
Muon
Electron
Charged Hadron (e.g. Pion)
Neutral Hadron (e.g. Neutron)
Photon
4T
2T
Silicon Tracker
Electromagnetic Calorimeter
Hadron Calorimeter
Superconducting Solenoid
Iron return yoke interspersed with Muon chambers
Transverse slice through CMS

Particle detectors



General overview of a HEP detector

Several layers made with different materials, electronics, readout

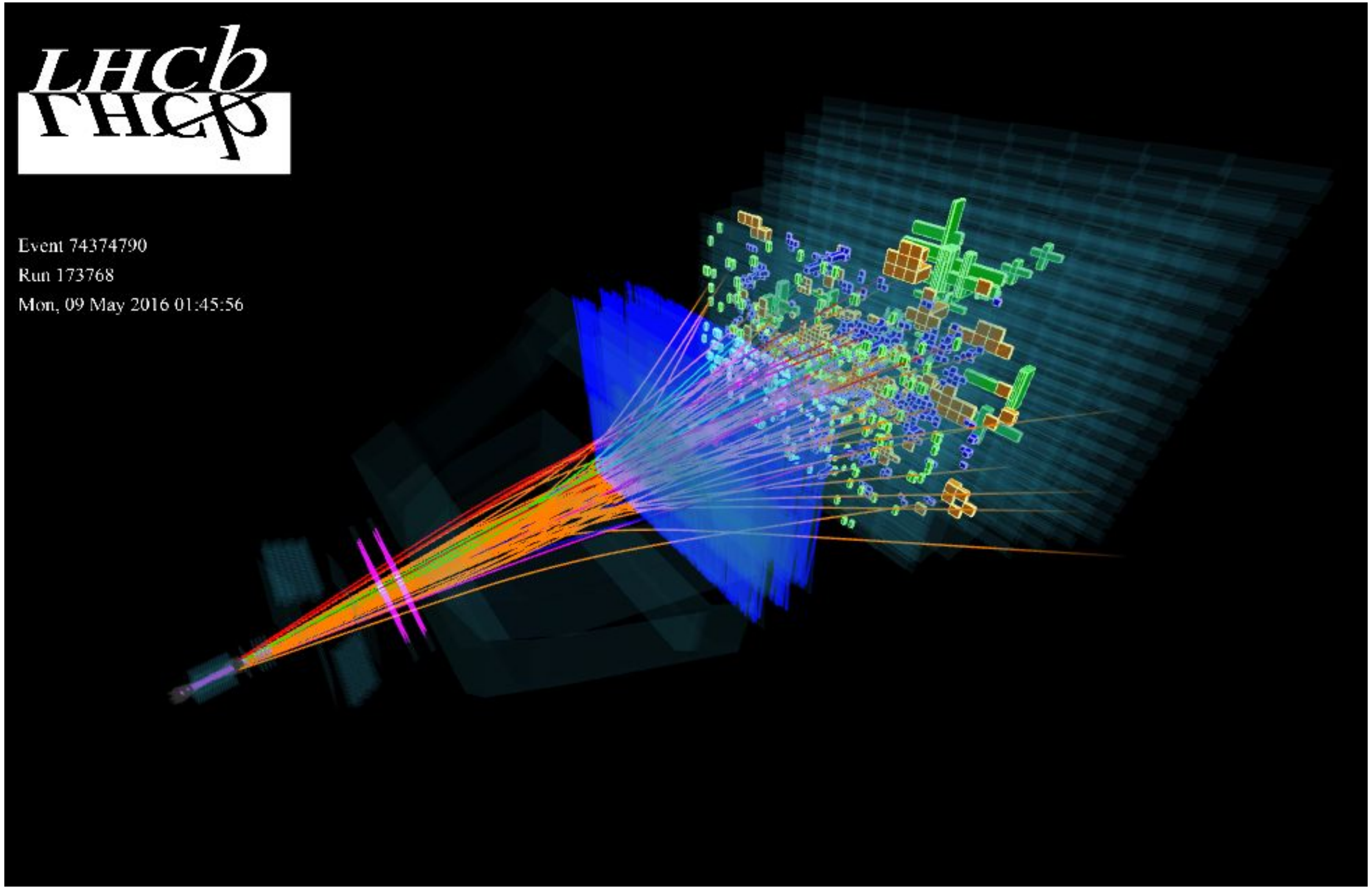
Each of them with a different goal

Different particle species interact with the materials differently

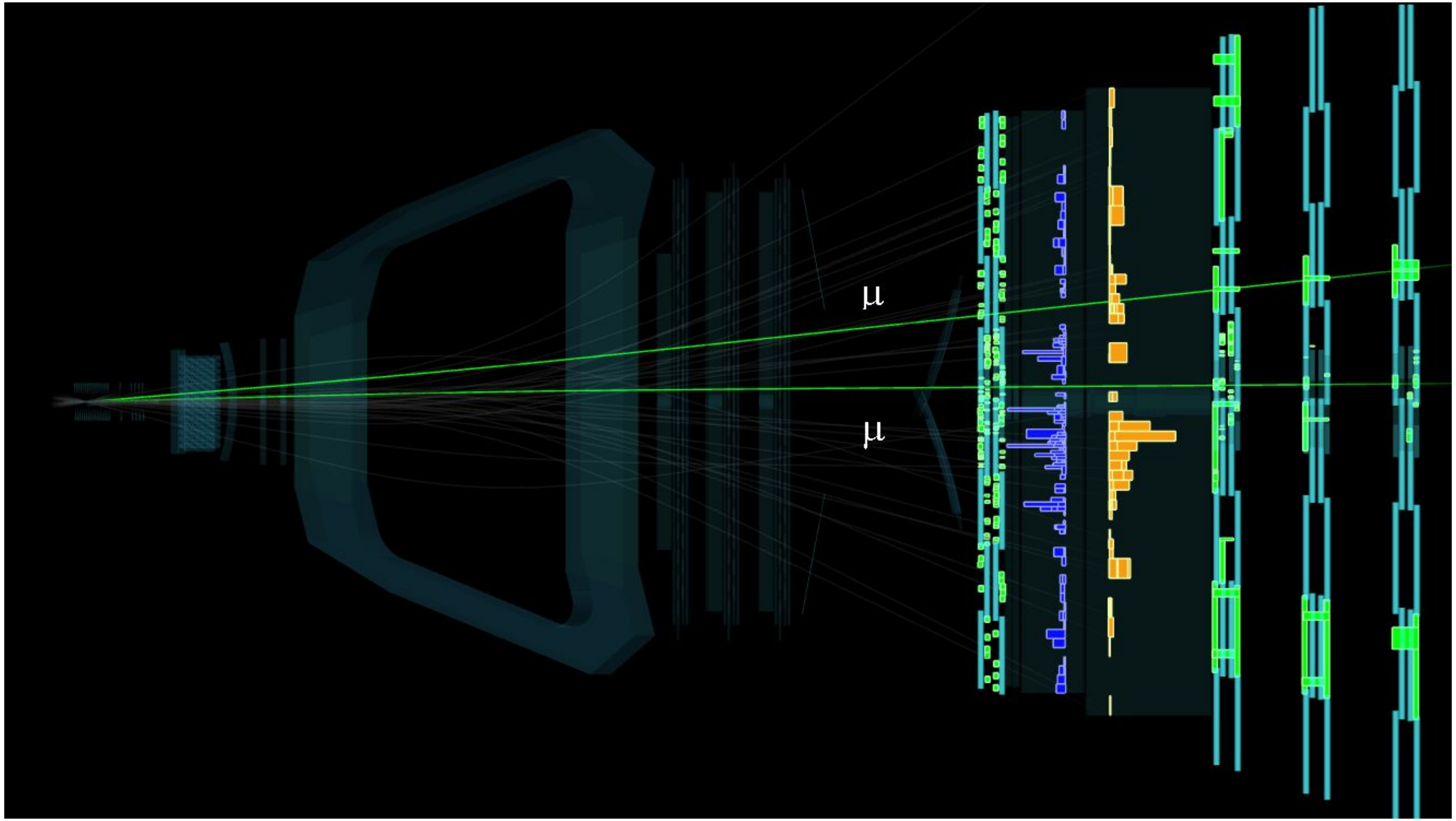
They all leave **hits** in the detector

From hits we reconstruct **tracks**

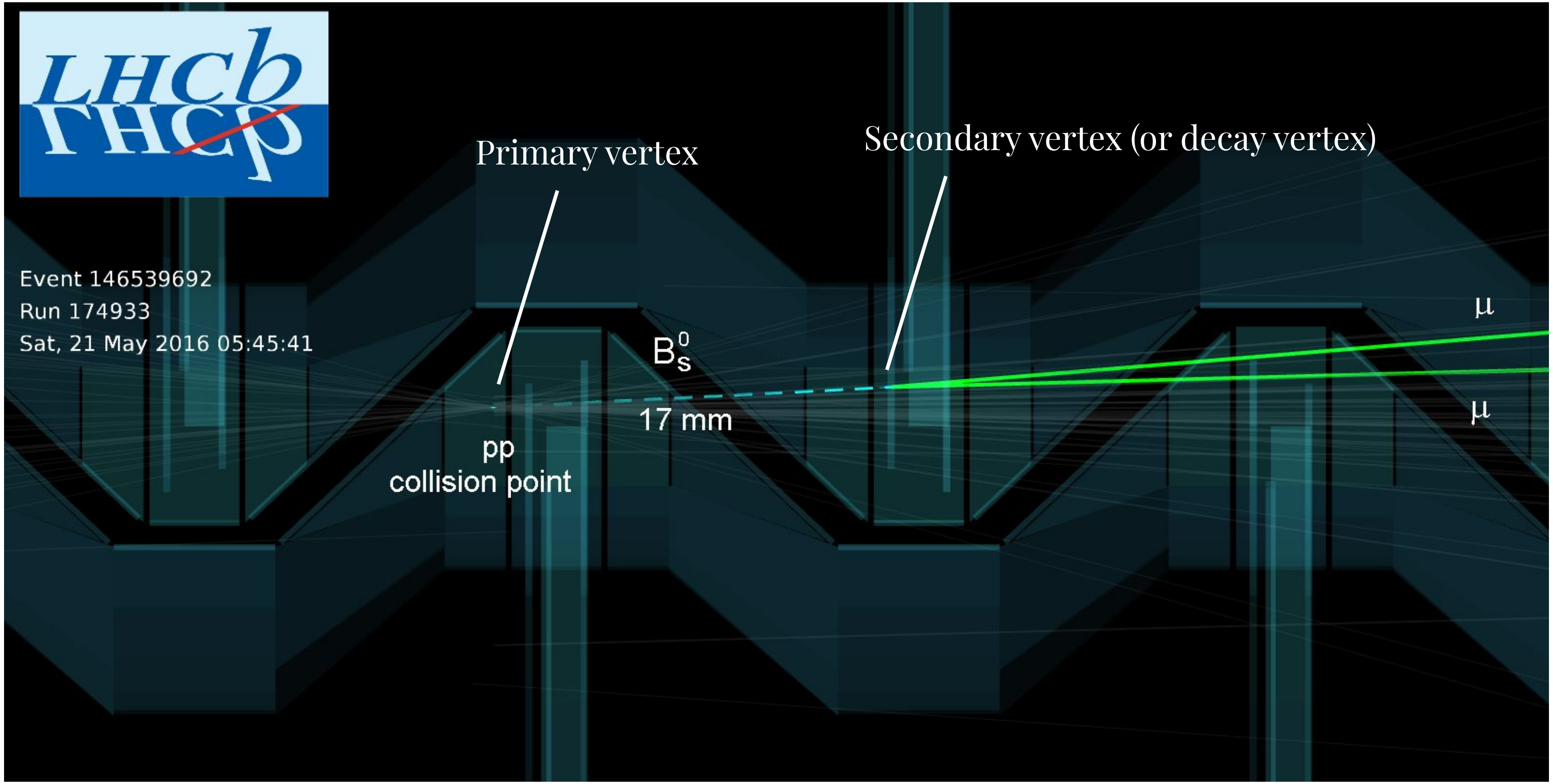
The LHCb detector at CERN



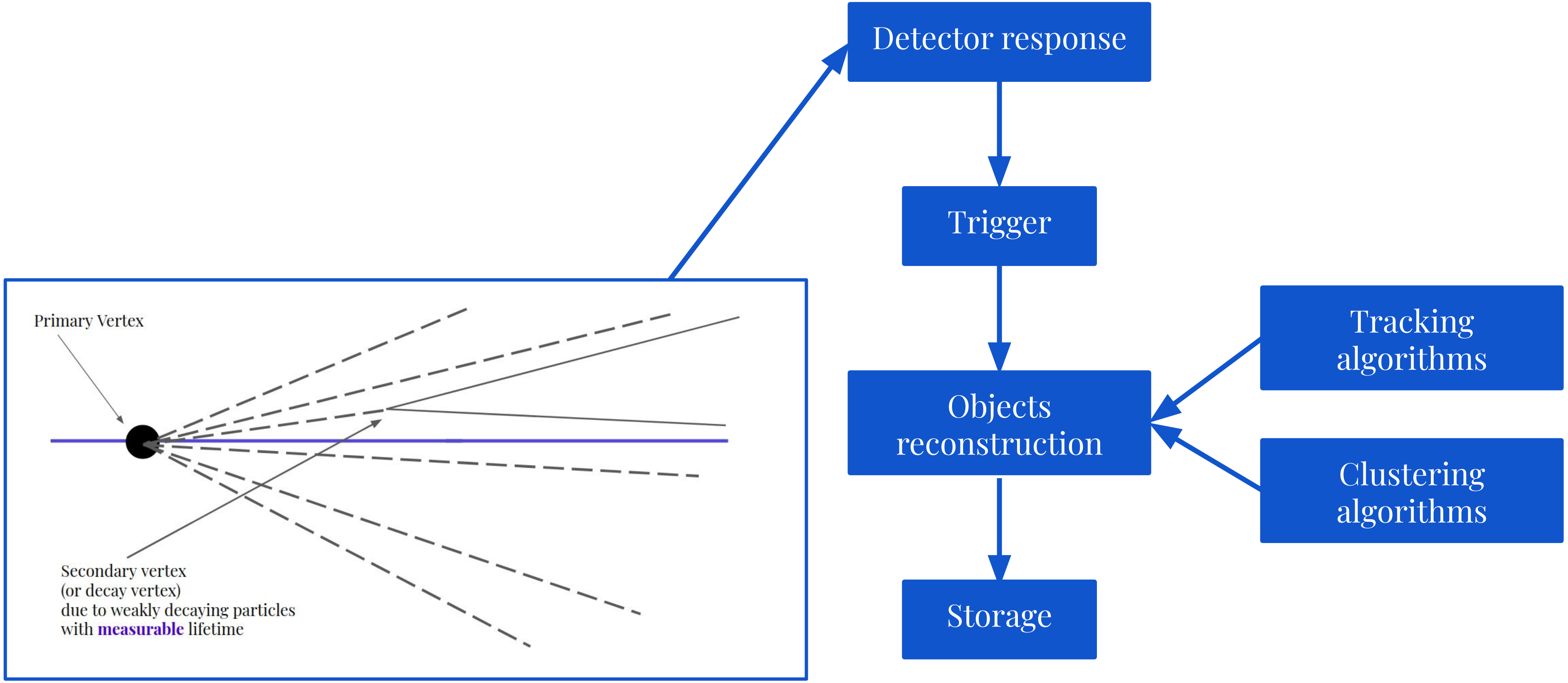
The LHCb detector at CERN



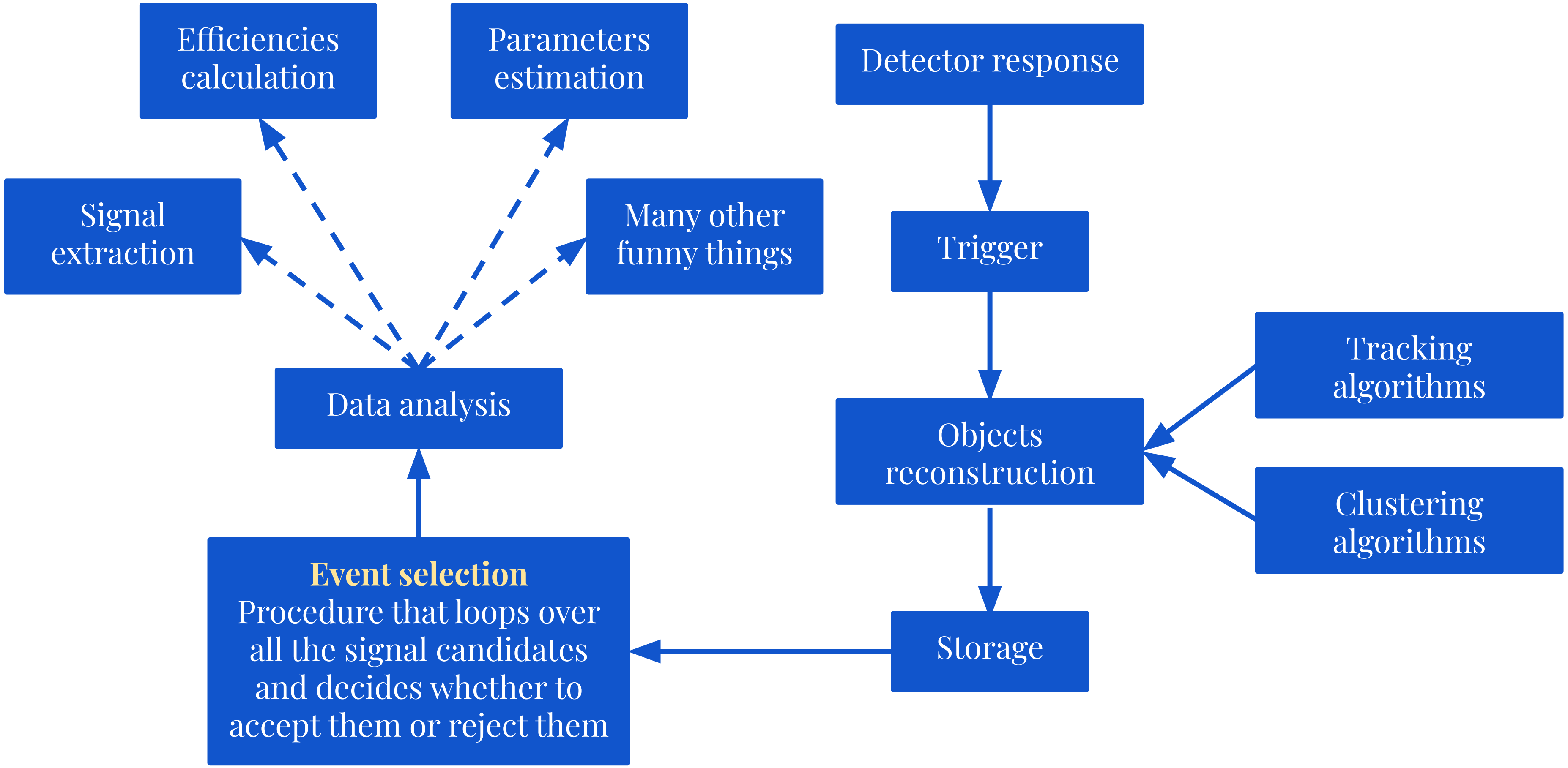
The LHCb detector at CERN



Typical workflow of a HEP experiment



Typical workflow of a HEP experiment



Signal efficiency and background rejection

Almost all datasets contain an **admixture of signal and background** which are selected by the trigger.
To tell apart the two categories we need to study their characteristics.
Two useful tools are the following:

- The **signal efficiency** of a selection is defined as $\varepsilon = S_f/S_o$, where S_f is the number of signal candidates after the selection and S_o is the number of signal candidates before the selection.

- The **background rejection** of a selection is defined as $R = (1 - B_f/B_o)$, where B_f is the number of bkg candidates after the selection and B_o is the number of bkg candidates before the selection.

S_f, S_o, and B_f, B_o are counted isolating regions where we have only signal or only background (it can be a simulated dataset - usually for signal - or a sideband - usually for background).

The perfect analysis is the one for which $\varepsilon = 1$ and $R = 1$ simultaneously for the full selection.

Signal efficiency and background rejection

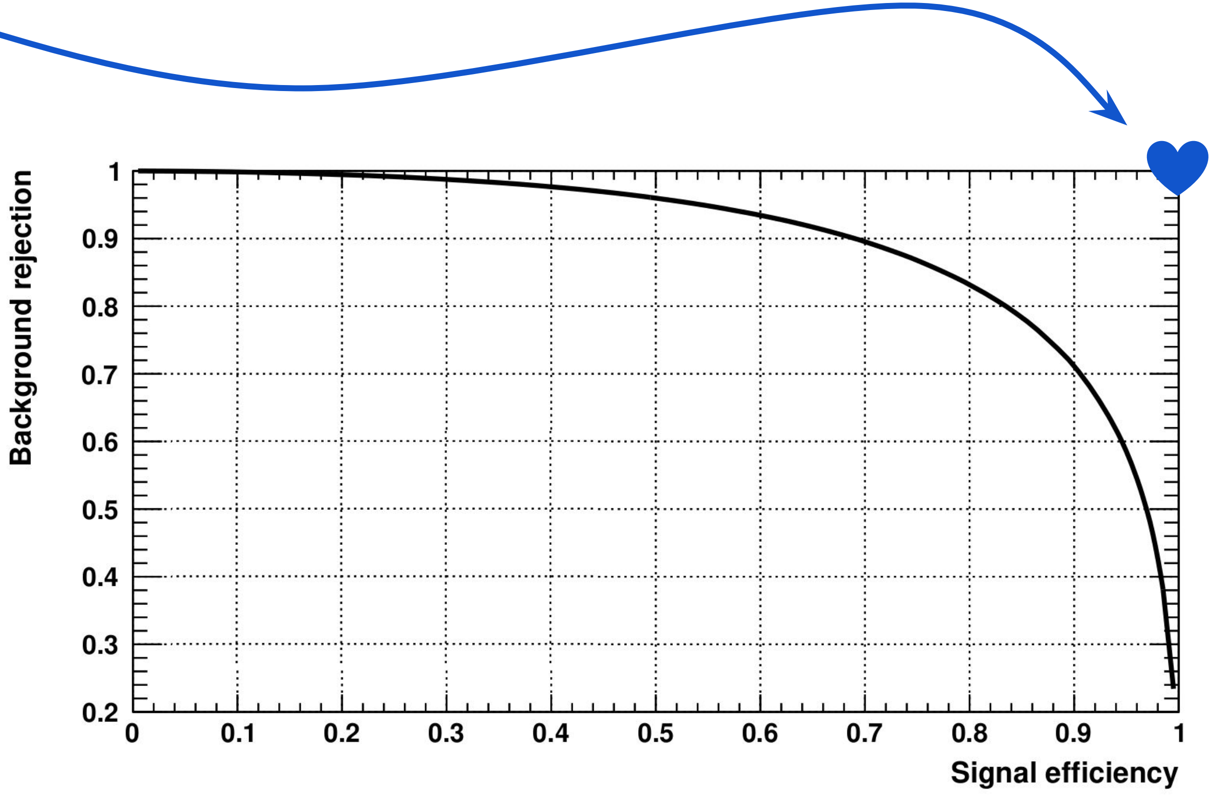
Of course, sadly the **perfect analysis** does not exist…

We must always find a compromise

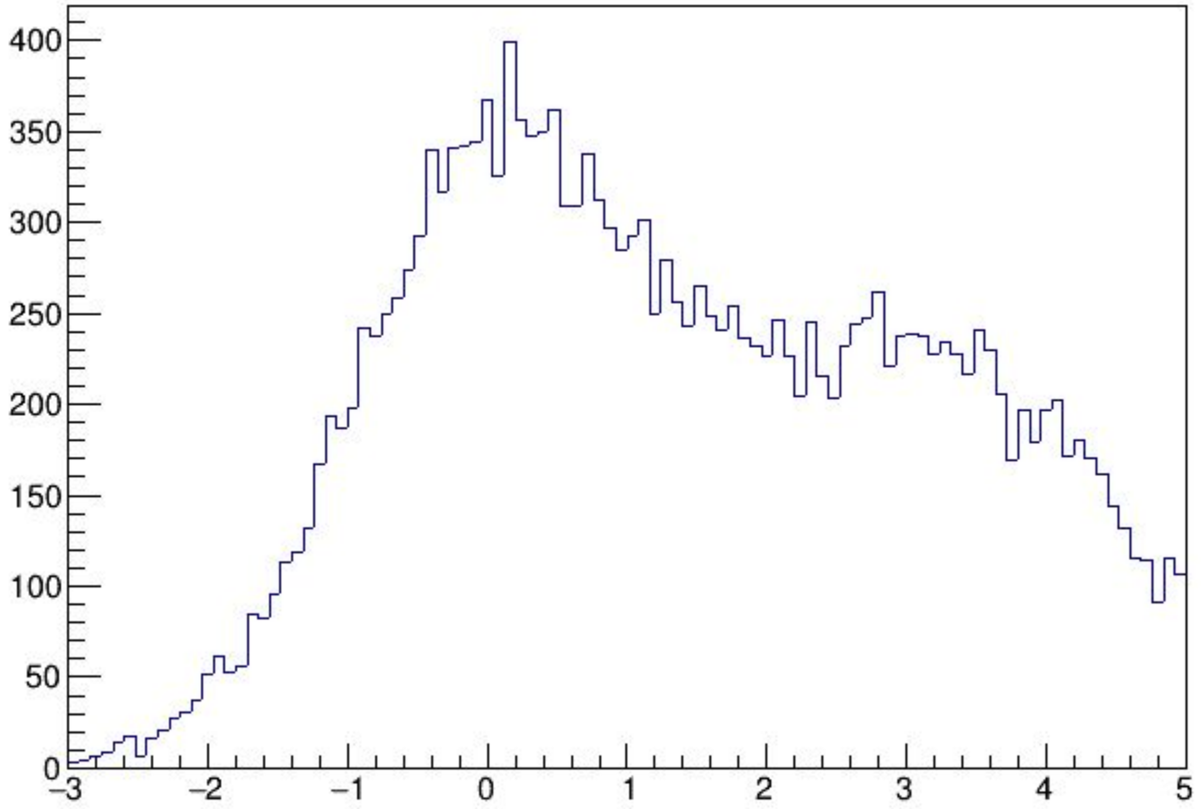
In special cases, one might choose “by eye” to exploit a certain feature

Usually there are better methods to choose the optimal point

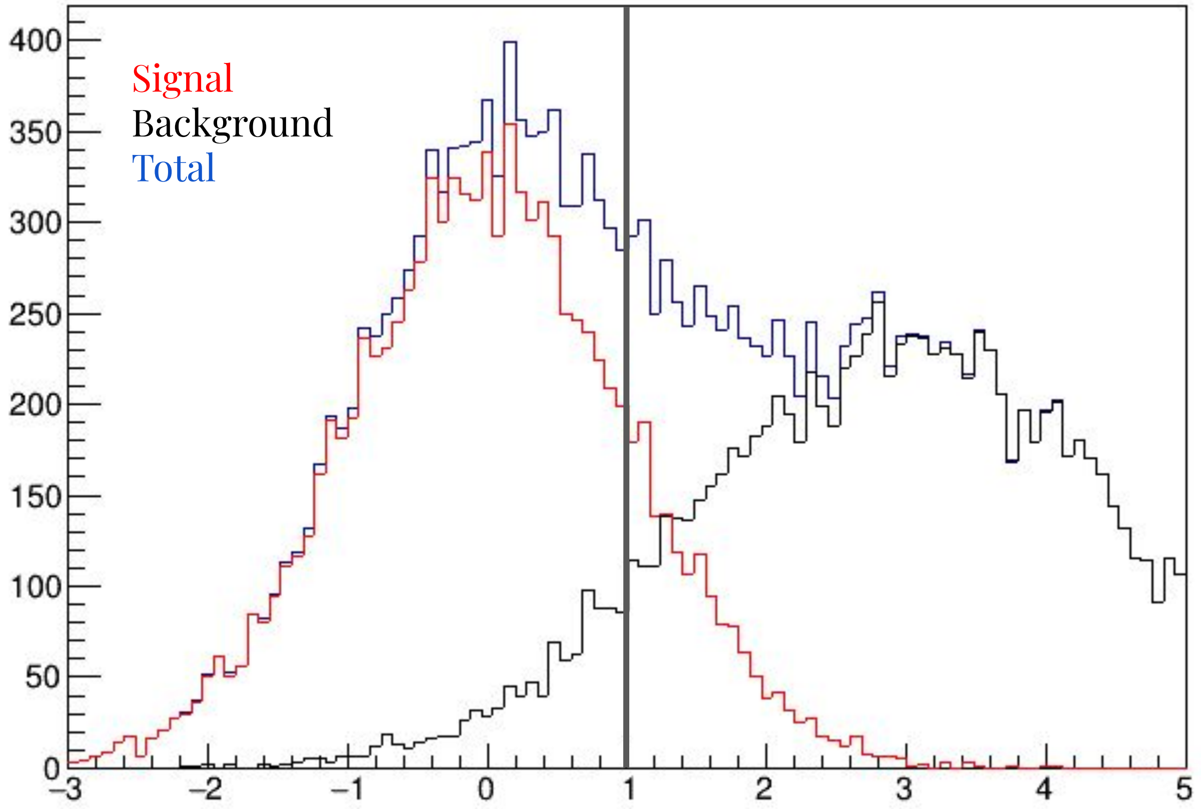
They depend mainly on what we are trying to measure



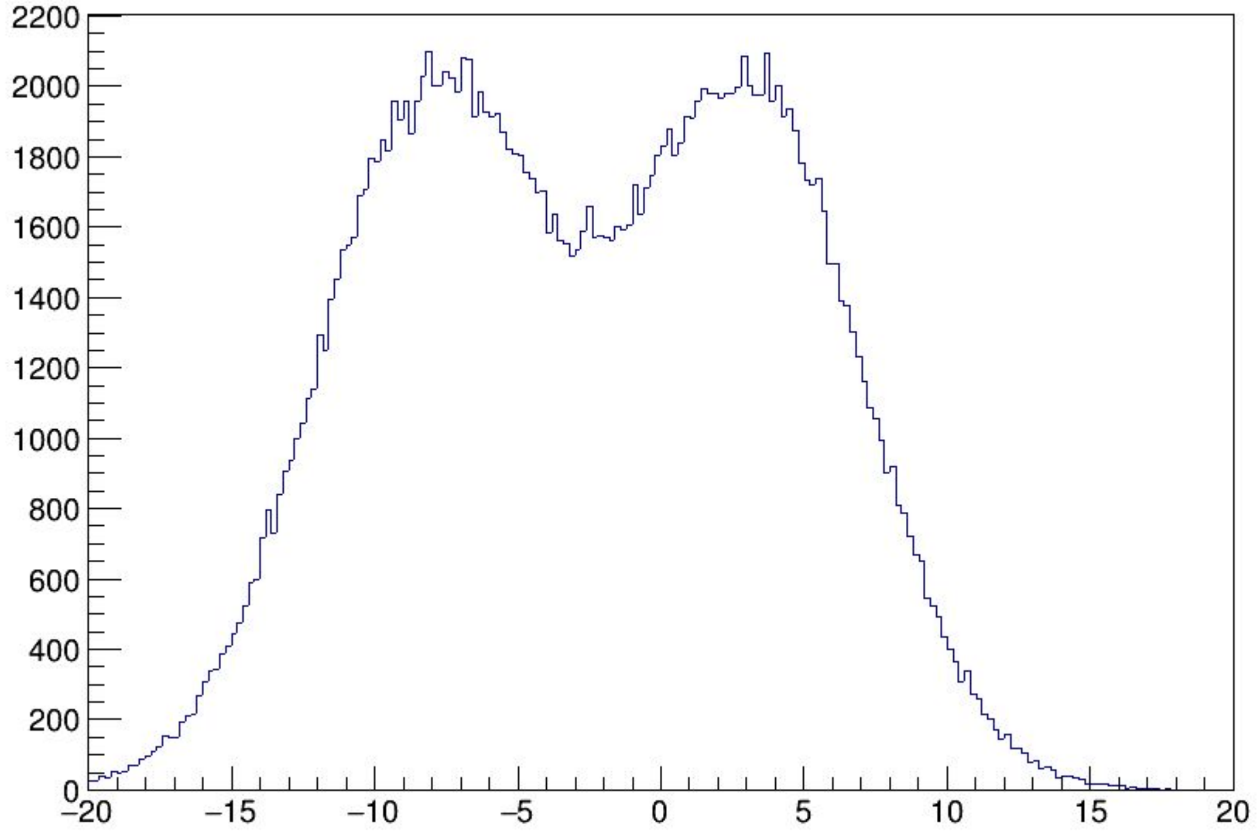
Cut-based selection and cut-flow



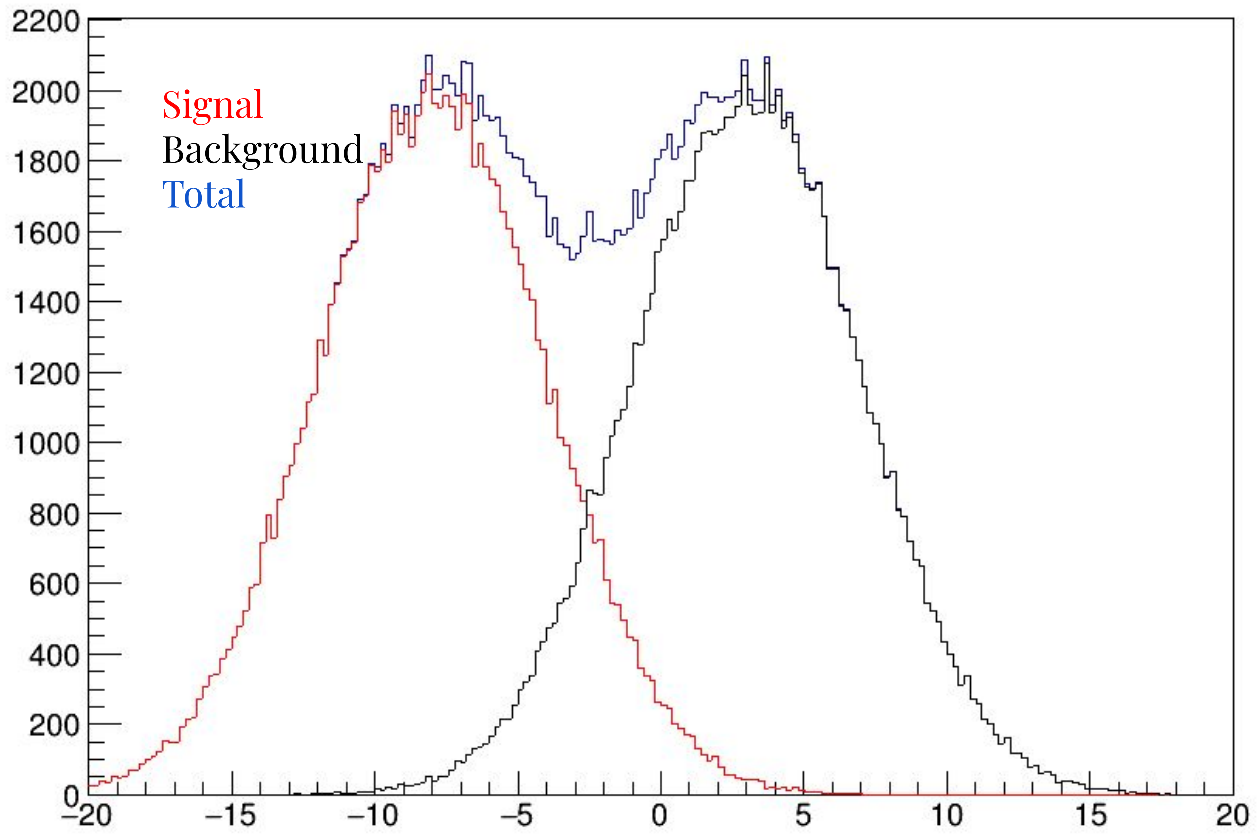
Cut-based selection and cut-flow



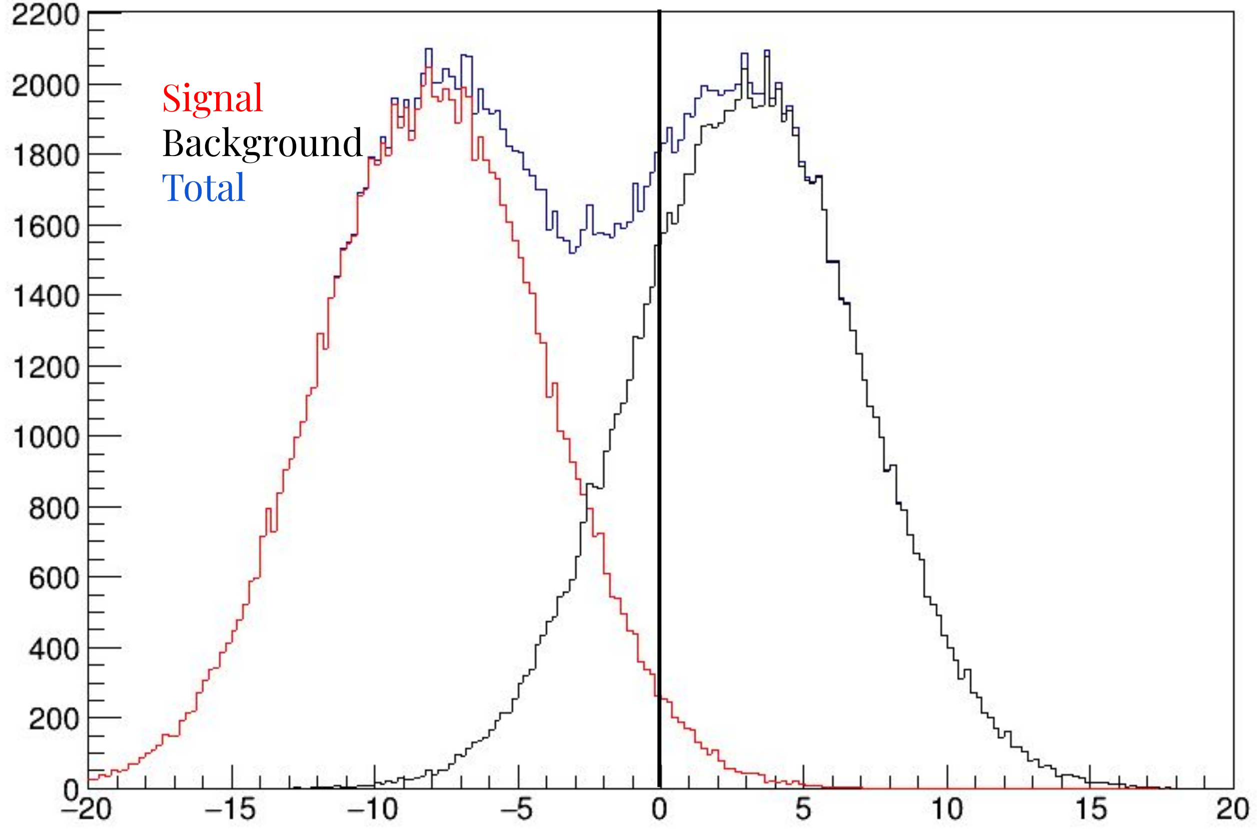
Bidimensional and multivariate selection



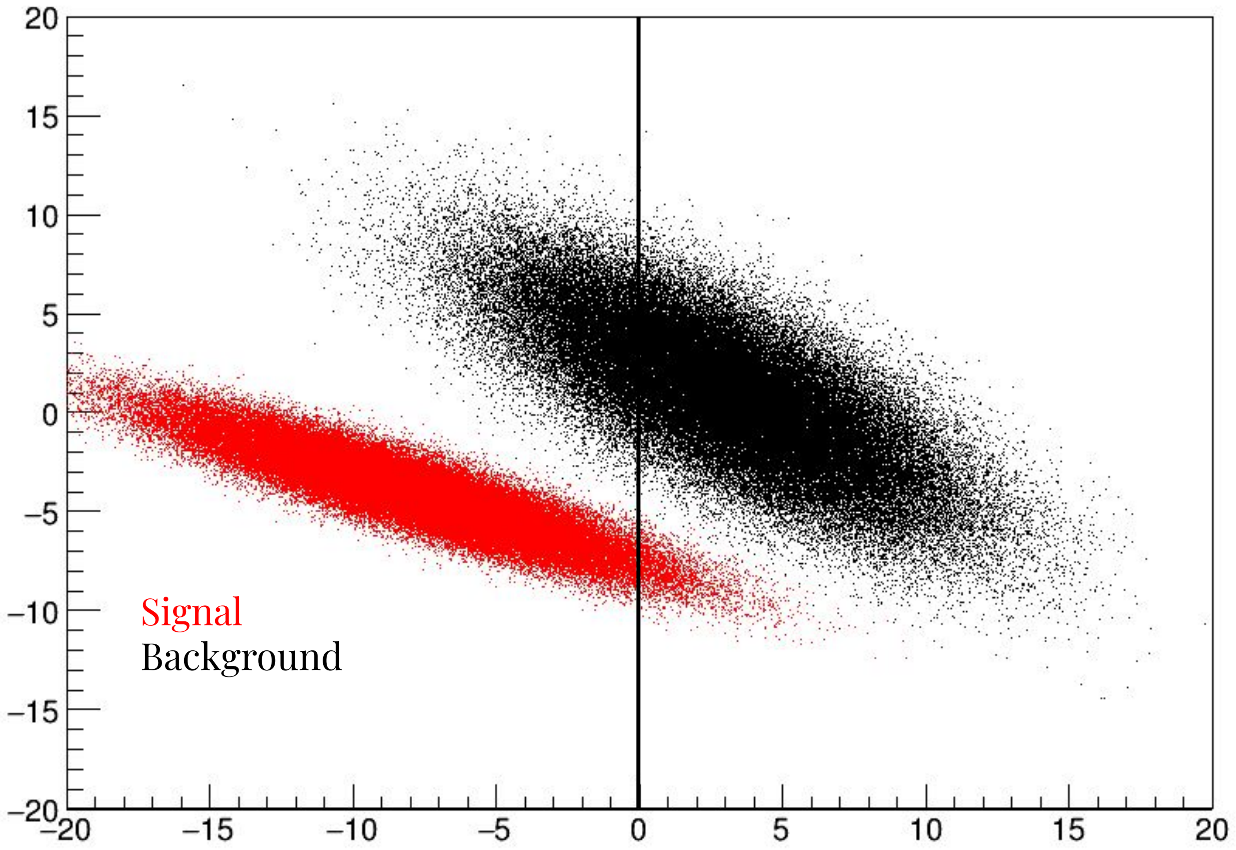
Bidimensional and multivariate selection



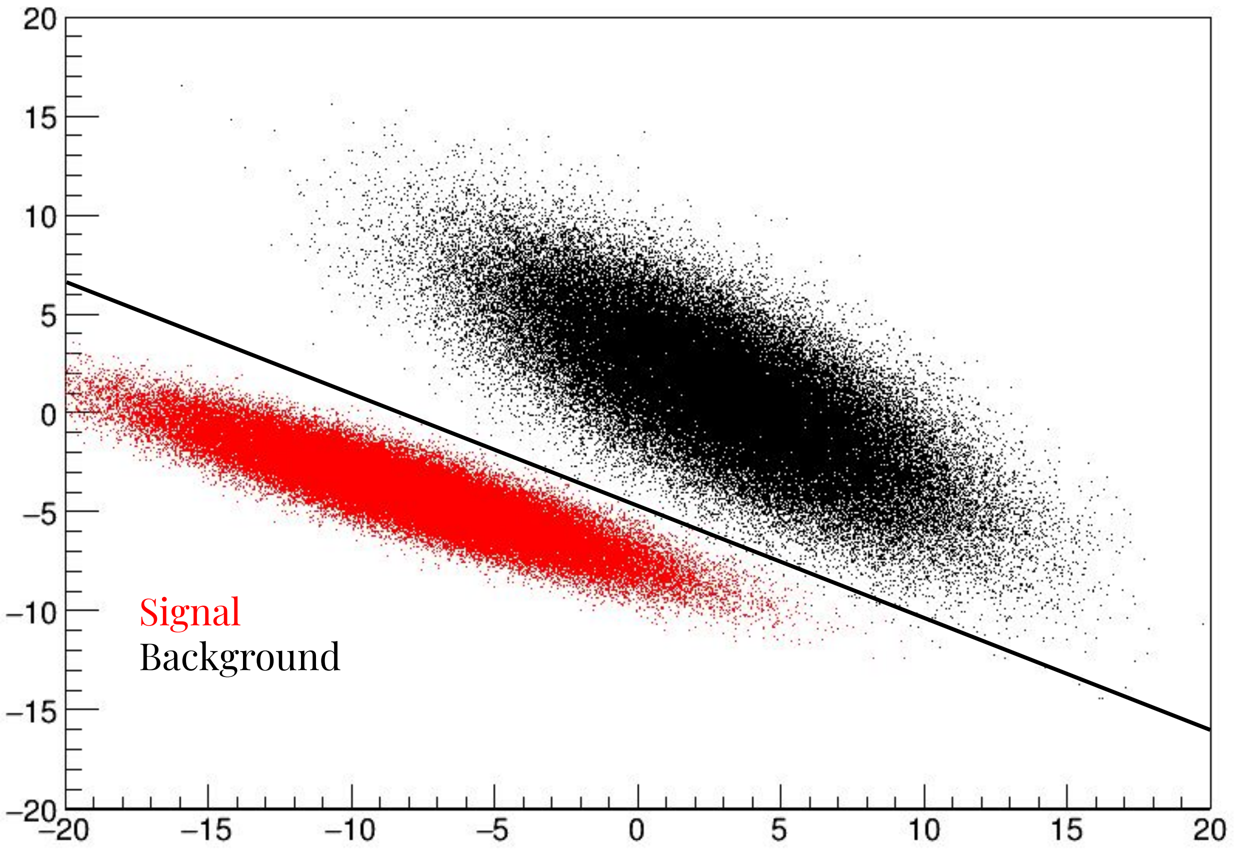
Bidimensional and multivariate selection



Bidimensional and multivariate selection



Bidimensional and multivariate selection



Bidimensional and multivariate selection

- In many cases, a cut-based selection is not enough
- If we look at the bidimensional histogram of two variables, they might have some correlation that we can exploit
- A better selection is a linear combination of the two variables, i.e. a diagonal cut in the 2D plane
- ...but it could also be a **nonlinear combination** (a curve in a 2D plane)
- In principle, one could extend this to a N-dimensional space...
- However, two major problems will ultimately lead to limitations to this method

Non-linear correlations between the discriminating variables

Increasing complexity and large number of combinations of variables

Score functions: statistical significance

In order to quantify the quality of a given cut or selection, we must first decide the best criterion. This depends on the process (observation, rare process, limit setting...).

This criterion is called a **score function** or, alternatively, a **figure of merit**.

In the most general cases (for instance, a measurement of a known quantity with a decently populated dataset) we can use the following.

Let's assume that, after applying a selection we are left with N candidates, which are the sum of S signal candidates and B background candidates:

$$S = N - B$$

Assuming that N is characterised by a Poissonian distribution, the uncertainty is:

$$\sigma^2(S) = \sigma^2(N) + \sigma^2(B) = N + \cancel{\sigma^2(B)}$$

in the hypothesis that we can estimate the value of B from MC or other sources with high precision and hence low uncertainty.

$$\frac{S}{\sigma(S)} = \frac{S}{\sqrt{N}} = \frac{S}{\sqrt{S+B}}$$

...i.e. **how many standard deviations is the signal away from zero**

Types of background

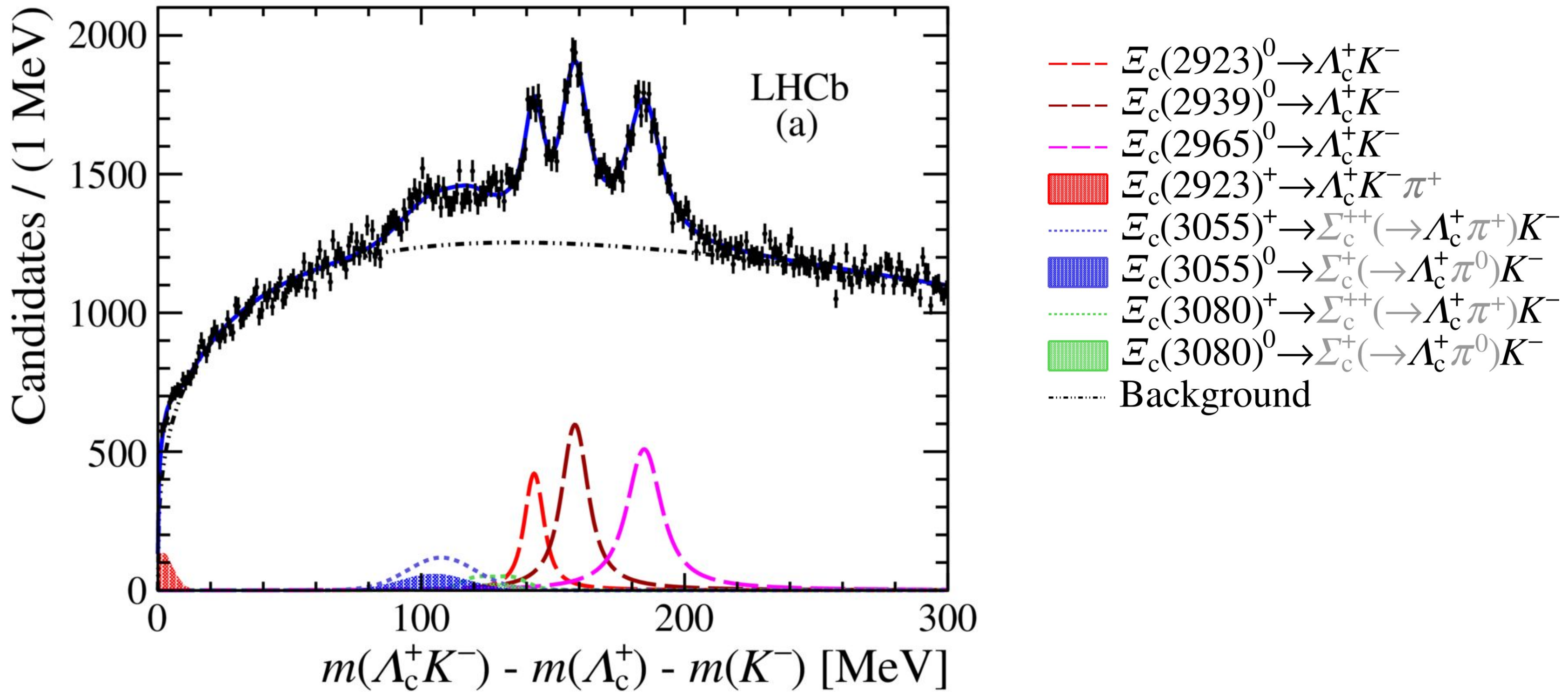
There are several types of background that can affect the data distribution and obscure the signal. Some of them are known and treatable; some are known but irreducible and some are unknown.

- **Combinatorial background**: in a proton-proton collision, hundreds of particles are created. When we select N tracks to form a decay chain, it is possible that some of them do not come from the decay we are interested in, but are instead just random tracks with similar kinematics and topology as the ones we are selecting.

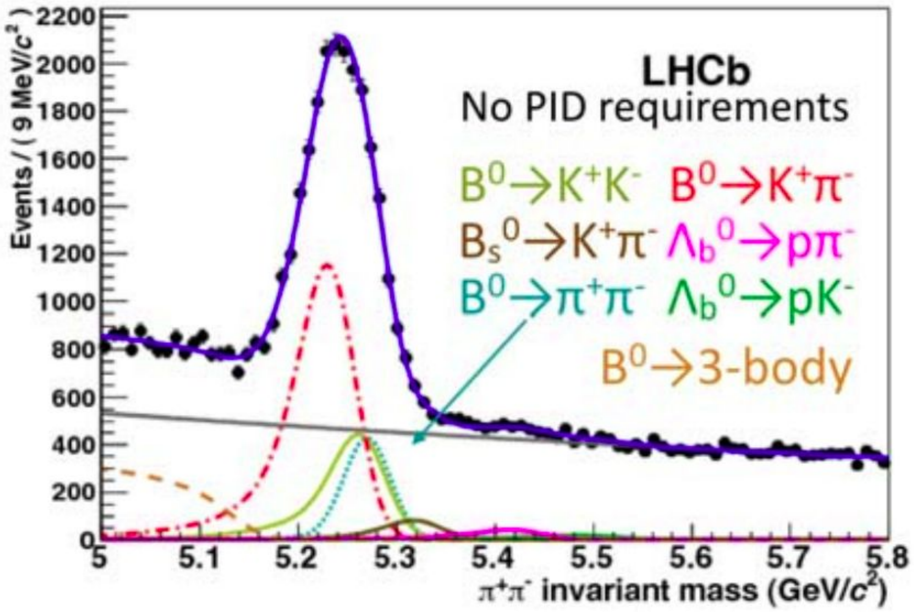
- **Partially-reconstructed background**: when selecting N tracks for a given decay chain, there is the possibility to include tracks which come from a more complex decay (with N+1, or even N+2 particles) which we are only selecting partially. The shape of this background will likely not be a nice peak. In the worst case scenario it must be studied with dedicated MC.

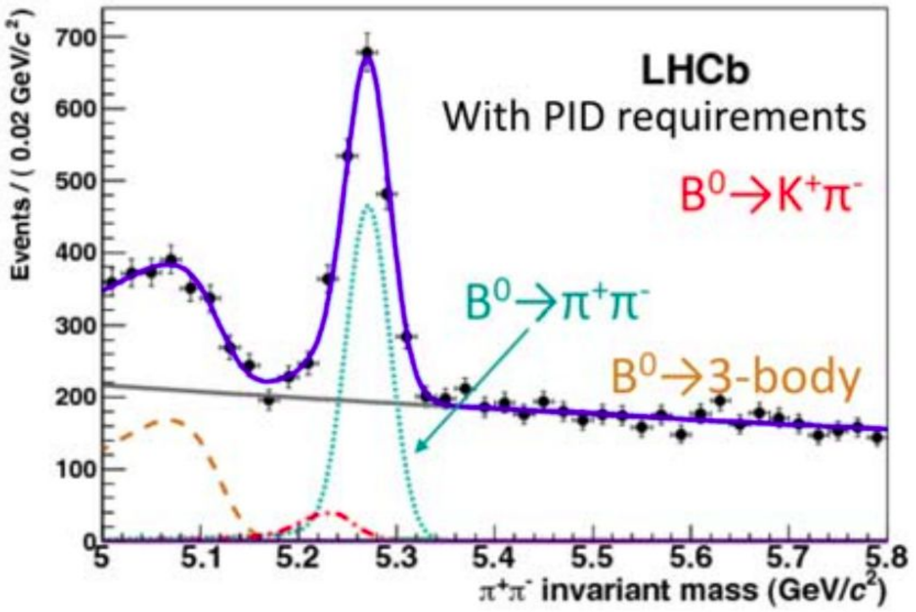
- **Misidentified background**: it can happen that our particle ID algorithms assign the wrong particle type to a given track. In this case, since the mass of the particle is assigned along with its type, then the energy of the track will be wrong as well and also the invariant mass of the combination of tracks. This background usually manifests itself as a (somewhat larger) peak with respect to the one we are interested in, at the wrong value of invariant mass.

Types of background (partially reconstructed)

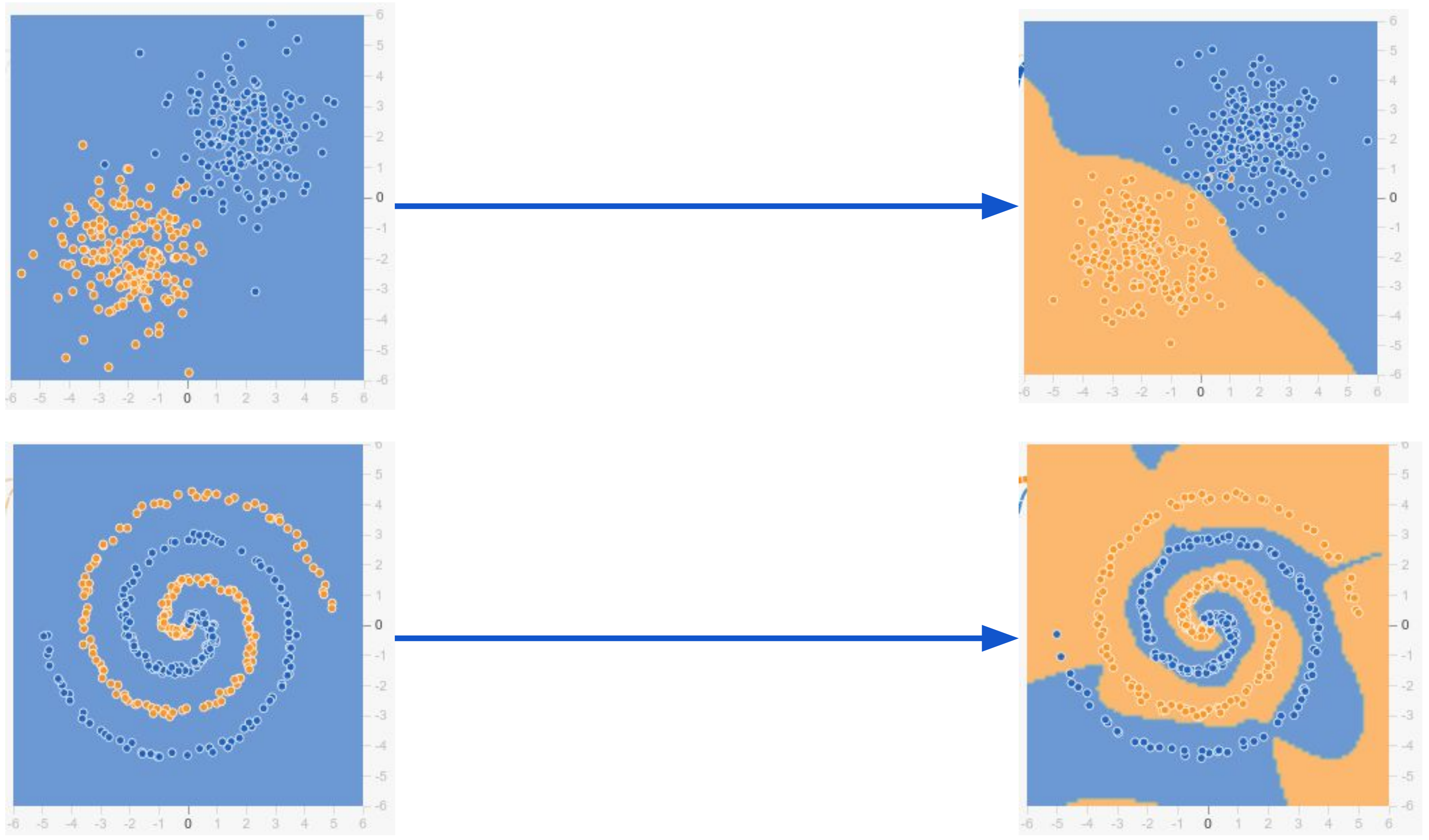


Types of background (combinatorial and misID)

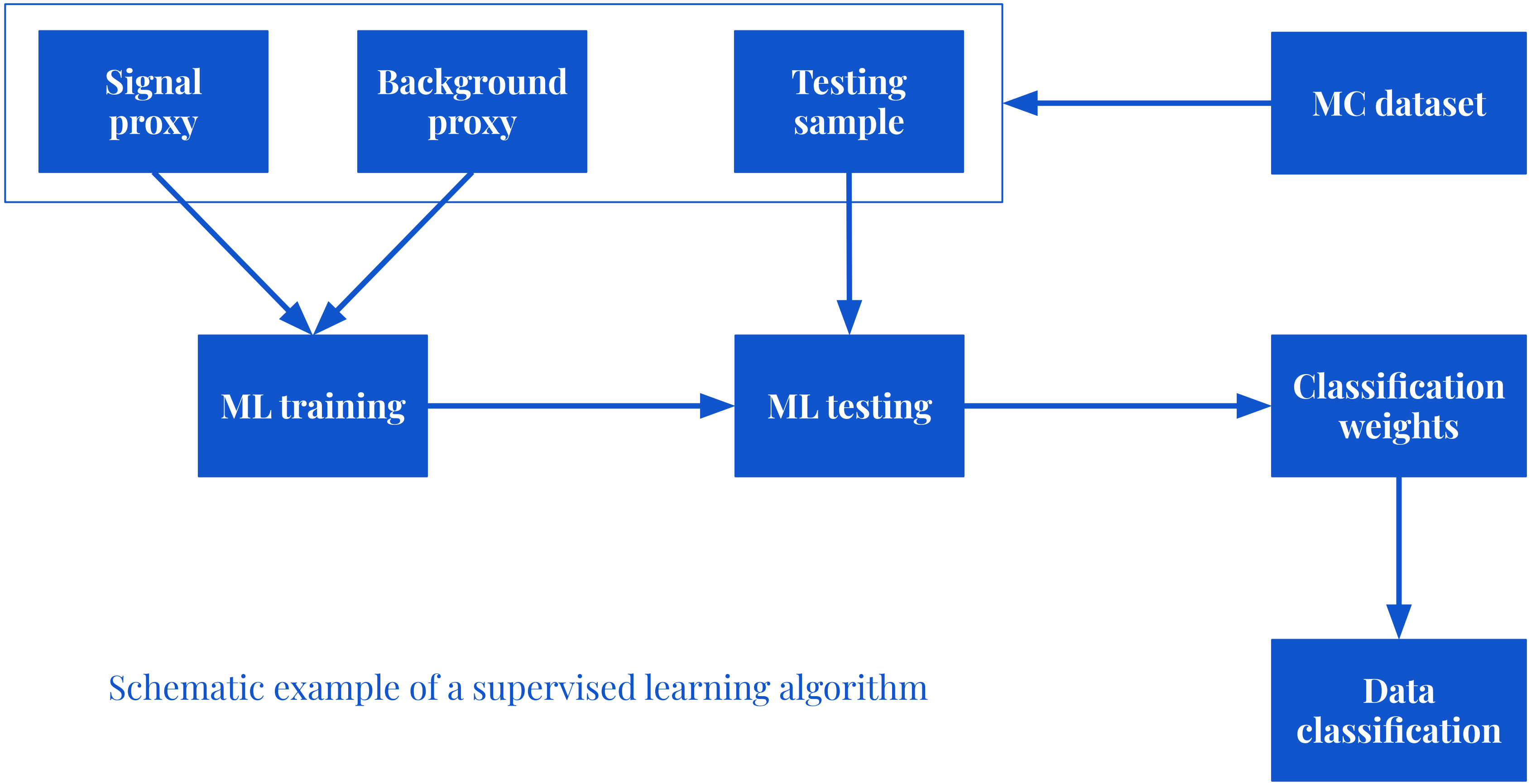




MVA and machine learning algorithms



MVA and machine learning algorithms



Schematic example of a supervised learning algorithm

Example of ML: artificial neural network (ANN)

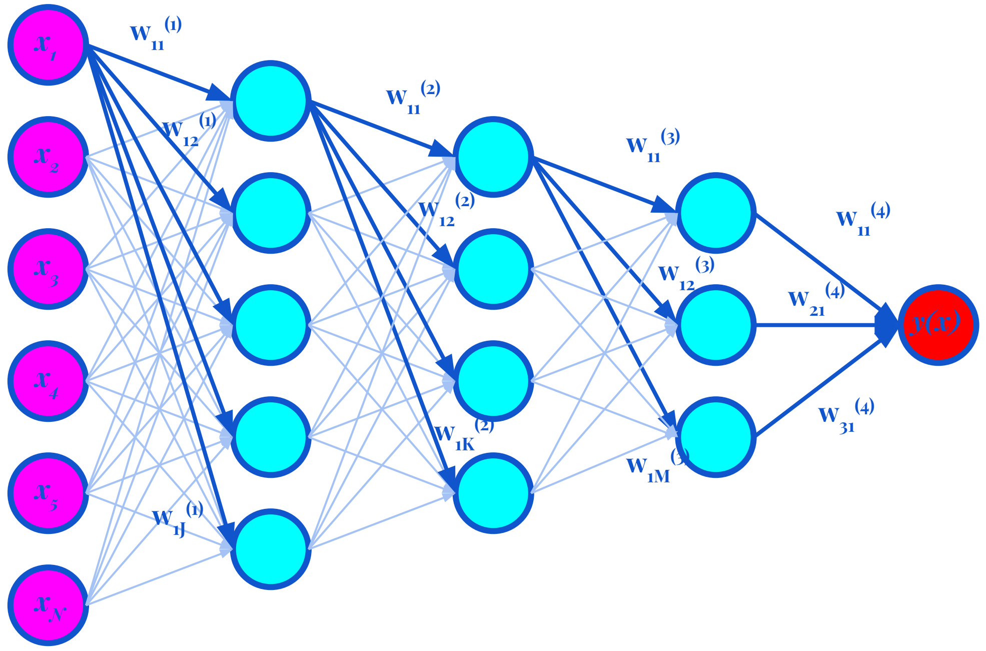
ANN are inspired by a (simplified) model of neuron cells in the human brain.

The output is computed combining the responses of multiple nodes.

The input nodes are arranged into the **input layer** and they take the data as input.

The output of the first layer is passed onto the second layer, and so on until the last **output layer**. All layers between the input and output are called "**hidden layers**".

The output of each node is computed as weighted sum of its input variables; the **weights** themselves are optimised during the training phase of the algorithm.



Example of ML: artificial neural network (ANN)

The training is achieved by minimising a loss function to get the weights

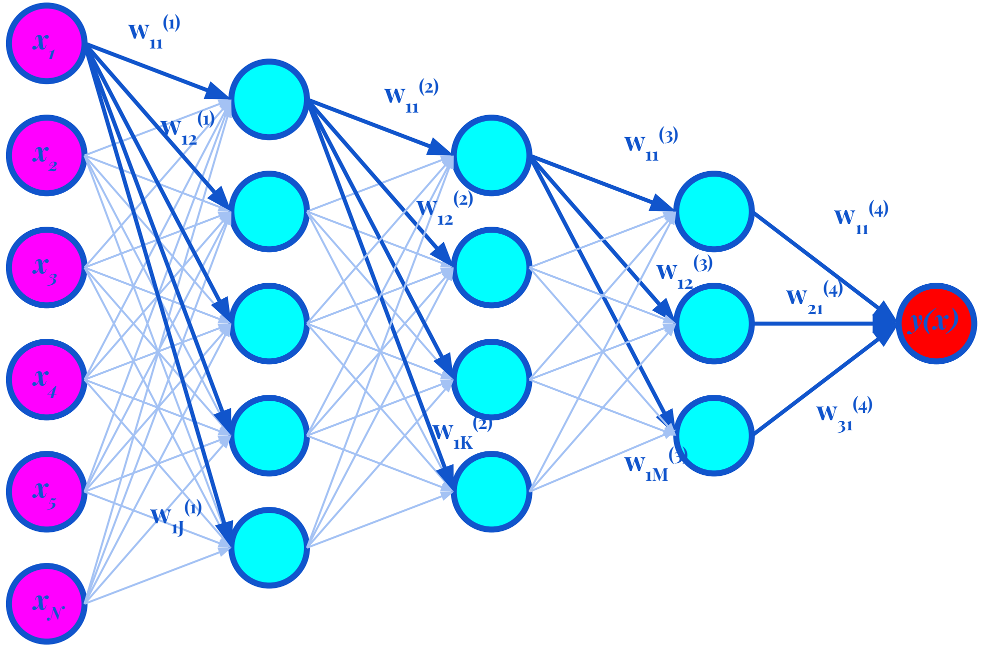
$$C(w) = \sum_{i=1}^{N} \left(y_i - \left(\sum_{j=1}^{K} w_{ij} x_j \right) \right)^2$$

and then calculating the output of the k-th node of the l-th layer as

$$y_k^{(l)}(\underline{x}) = \varphi \left(\sum_{j=1}^{n_l} w_{kj}^{(l)} x_j \right)$$

where $\varphi(z)$ is the **node activation function**, often taken as a sigmoid:

$$\varphi(z) = \frac{1}{1 + e^{-\lambda z}}$$



Example of ML: boosted decision tree (BDT)

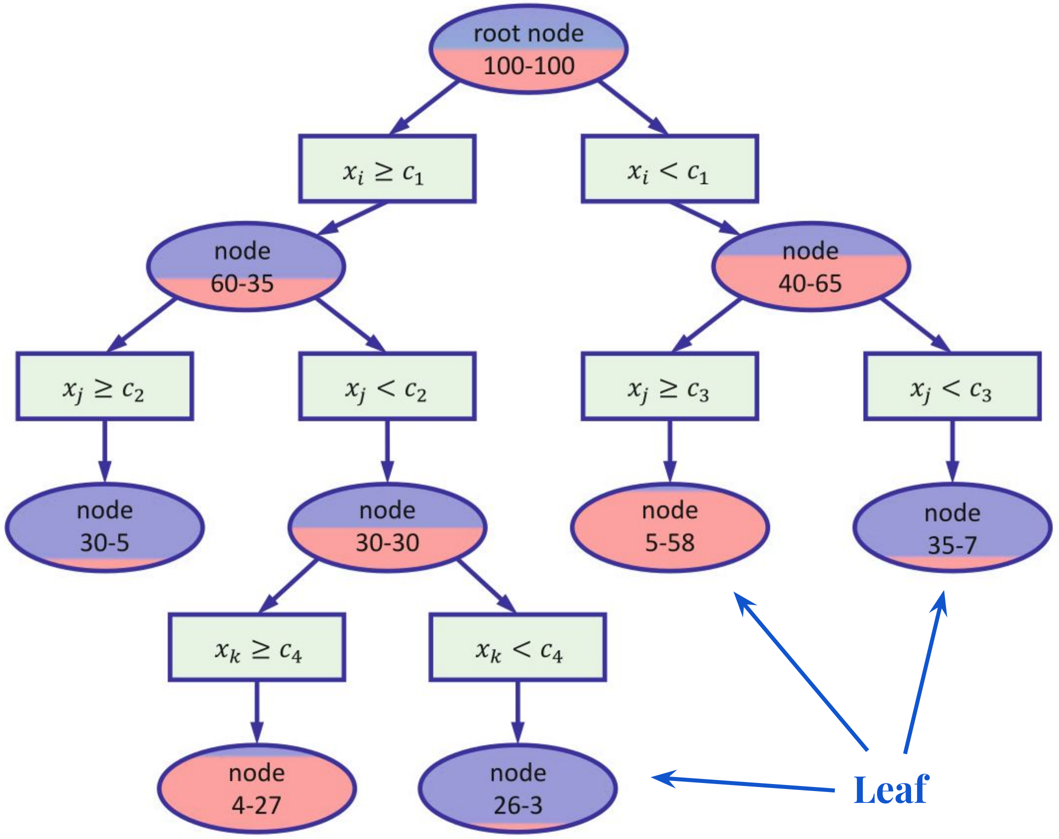
BDT is one of the most used ML algorithms in HEP (but not necessarily the best)

A decision tree is a subsequent series of cuts on randomly selected variables. After a certain number of steps, it stops and it classifies the selection as signal S or background B.

The cuts are chosen to optimise a FoM.

This is an example of weak classifier, as a single tree is basically an automated procedure to apply a series of cuts.

In order to boost the performances we need a higher level of complexity.



Example of ML: boosted decision tree (BDT)

This is achieved by the so-called **Random Forest**.

A large number of trees contribute to the final output. For each input event, the output of all trees is considered and the final decision is calculated as the average of the single ones.

We could use the information of a tree into the training of another tree.

We can, therefore, create a forest by iteratively adding new trees which are optimised based on the decisions obtained in the previous iteration.

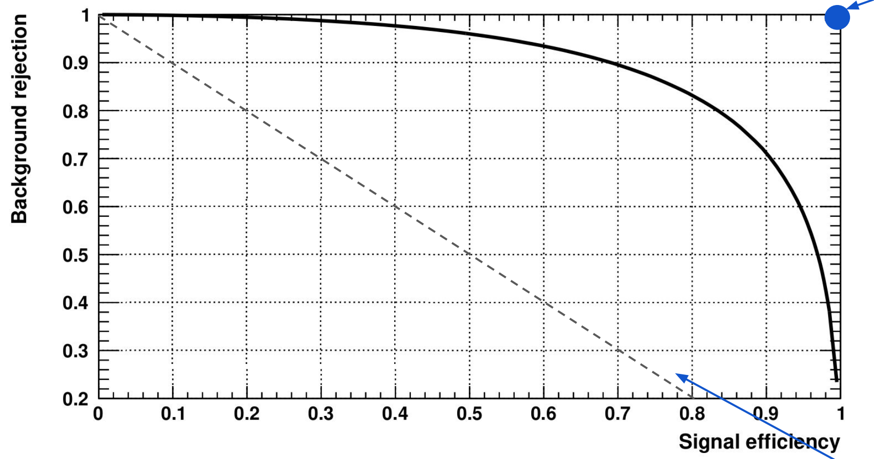
This procedure is called **boosting**.

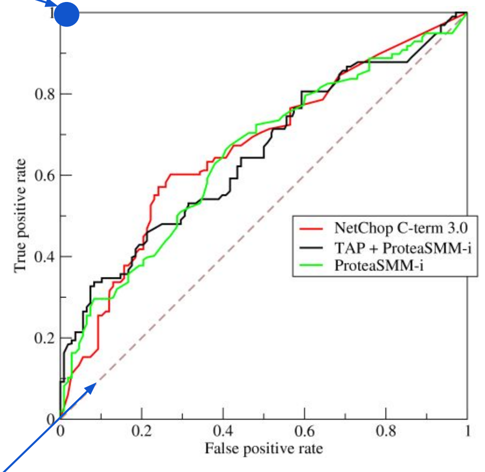
A Random Forest combines the votes of all trees

ROC curve and AUC discriminant

The ROC (Receiver Operating Characteristic) curve is a graphical system to assess the quality of a classifier and they are largely used with machine learning algorithms.

Perfect classifier





Random classifier

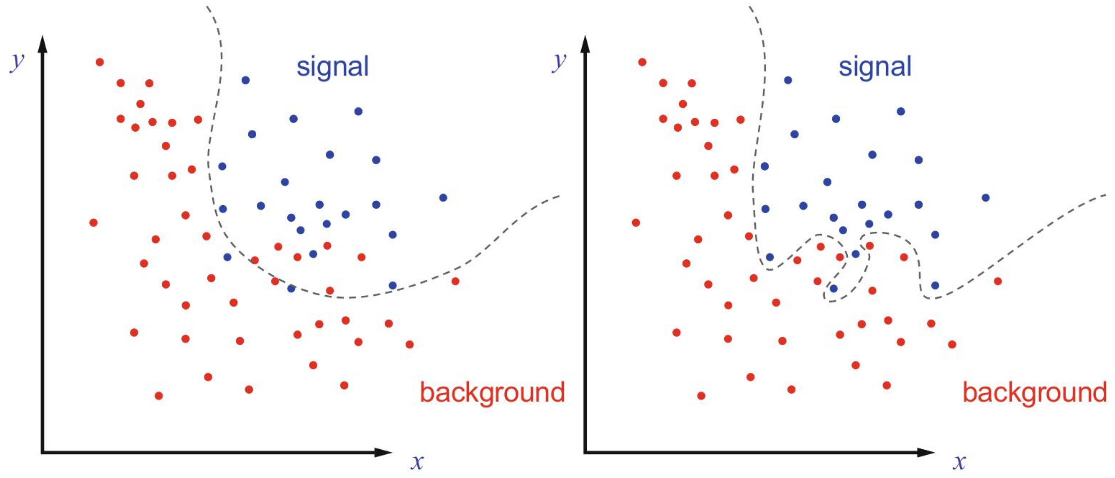
The area under the curve (AUC) is often used as discriminant to select which classifier works best in the specific situation

Undertraining and overtraining

Any ML algorithm suffers from two general potential problems:

- **Undertraining**: the size of the training sample is not large enough to properly populate the hyperspace of the input variables, but the algorithm parameters are tuned to reflect this issue. In this case, **the algorithm performances will be sub-optimal**.

- **Overtraining**: if the parameters of the algorithm are tuned in an inadequate way with respect to the sample size, the training procedure might exploit artificial structures due to statistical fluctuations which are not representative of the expected distributions. This results in a **wrong classification** of signal and background events.



The Kolmogorov-Smirnov test

One way to assess the presence of overtraining in a training procedure is to use the Kolmogorov-Smirnov test: in order to do so, any MVA framework will split the input sample into a *training* sample and a *testing* sample.
After performing the training with the relative sample, the output weights are applied to the testing sample and the two distributions are compared.

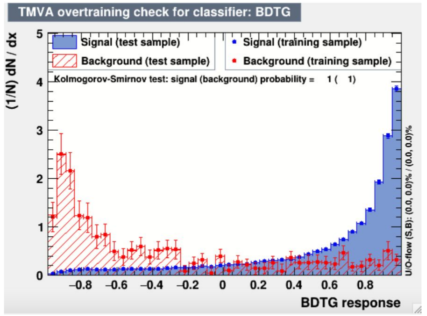
The Kolmorov-Smirnov test calculates the maximum distance ***D****, over all bins, between the cumulative distributions of the multivariate algorithm output for the training set and the testing set, separately.

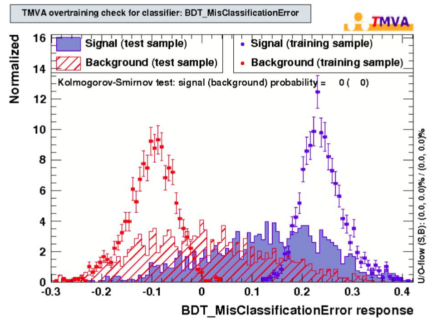
This quantity is a random variable with an asymptotically uniform distribution between 0 and 1, after an opportune transformation is applied.

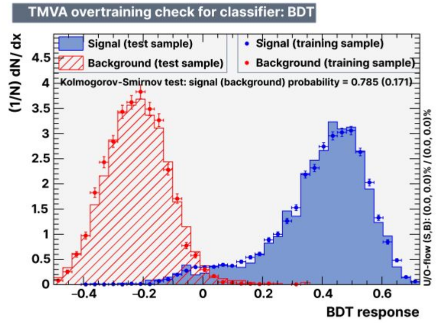
Then, a set of *N* pseudoexperiments is generated from the training and testing distributions, and the test is repeated for all of them, obtaining each time a distance $\boldsymbol{D_n}$.

The Kolmogorov-Smirnov probability p_{KS} is then defined as the number of times that $\boldsymbol{D_n > D^*}$, divided by *N*.

If the two distributions are similar, then on average the fluctuations of the pseudoexperiments will be larger than the first case about half of the time. So if (p_{KS}≃0) or (p_{KS}≃1), then it is likely that the algorithm has been overtrained.


TMVA overtraining check for classifier: BDTG
Signal (test sample)
Background (test sample)
Signal (training sample)
Background (training sample)
Kolmogorov-Smirnov test: signal (background) probability = 1 (1)
(1/N) dN / dx
BDTG response
U/O-flow (S,B): (0.0, 0.0)% / (0.0, 0.0)%


TMVA overtraining check for classifier: BDT_MisClassificationError
TMVA
Signal (test sample)
Background (test sample)
Signal (training sample)
Background (training sample)
Kolmogorov-Smirnov test: signal (background) probability = 0 (0)
Normalized
BDT_MisClassificationError response
U/O-flow (S,B): (0.0, 0.0)% / (0.0, 0.0)%


TMVA overtraining check for classifier: BDT
Signal (test sample)
Background (test sample)
Signal (training sample)
Background (training sample)
Kolmogorov-Smirnov test: signal (background) probability = 0.785 (0.171)
(1/N) dN / dx
BDT response
U/O-flow (S,B): (0.0, 0.0)% / (0.0, 0.0)%