Application of computational intelligence in analysis of heavy hadrons decays

AGH Summer School of Physics 2020

Author:
Jerzy Pryga

Supervisor:
mgr Wojciech Krupa

July - September 2020

What does the LHCb experiment produce?

First of all it produces **A LOT OF DATA**
(by A LOT we mean really huge amount for which we need some clever methods of analysis).

This data contain many variables which describe features of products of the B meson decays [1]. Our processes:

Primary decay:

Without photon emission:

$$B_s^0 \to D_s^{\mp} K^{*\pm}$$

With photon emission:

$$B_s^0 \to D_s^{*\mp} K^{*\pm}$$
$$D_s^{*\mp} \to D_s^{\mp} \gamma$$

Secondary decays :

$$D_s^{\mp} \to K^+ K^- \pi^{\mp}$$
$$D_s^{\mp} \to \phi \pi^{\mp}$$
$$D_s^{\mp} \to K^{*0} K^{\mp}$$

$$K^{*\mp} \to K_s^0 \pi^{\pm}$$
$$K_s^0 \to \pi^+ \pi^-$$

What are we looking for?

Detectors measure:

Masses of particles:

$m_1,\ m_2,\ m_3\ldots$

Momenta of particles:

$p_1,\ p_2,\ p_3\ldots$

Energy of a single particle: $E = \sqrt{m^2 + \vec{p}^2}$

Mass of a two particle system: $m_{12} = \sqrt{(E_1 + E_2)^2 - (\vec{p_1} + \vec{p_2})^2}$

Knowing masses and momenta of the products of the decay we can obtain m_{12} distribution.

Getting rid of the background shows us shape we expect, and so we can identify the primary particle (our job).

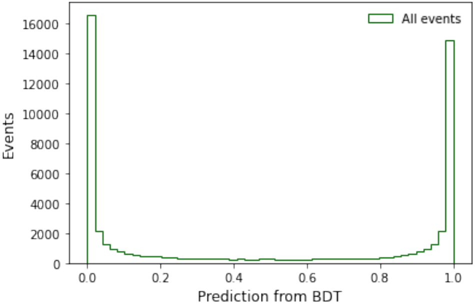
Used tools - XGBoost

Tool used to perform this task is called **XGBoost** [2, 3]. It is a library which:

1. trains algorithms,
2. implements Gradient Boosting machine learning technique,
3. is designed to be effective, flexible and portable,
4. solves many different data science problems in an accurate and fast way,
5. is compatible with programming languages like Python, R, Julia and Scala.

XGBoost BDT prediction

The goal is to distinguish **background signals** and **event signals** (binary logistic function). The output of classifier is BDT prediction which is calculated for each event.



BDT predictions is a probability that given signal belongs to one of two different types - in our case:

~ 1 - interesting event, ~ 0 - background event,

What we did?

Steps of our work:

1. Creating a classifier.
2. Choosing classifier parameters.
3. Training it:
   1. using Monte Carlo generated events as training data - label: **signal** / **background** is known,
   2. splitting data into two sets: training set and testing set,
   3. setting parameters, choosing classificatory variables etc.,
   4. training classifier,
   5. testing it on testing set,
   6. estimating efficiency of the classifier (we have label) - if it is unsatisfactory go to step 3 and repeat.

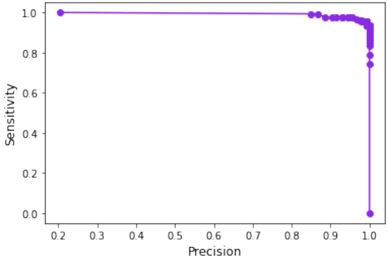
Rating classifier performance - ROC curve

The most important parameter which describe the efficiency of our classifier is the **ROC curve**. For ROC curve:

1. $\uparrow$ Sensitivity.
2. $\longrightarrow$ Precision.
3. Points every 0.01 value of BDT prediction value (from 0 to 1).

$$\text{Sensitivity} = \frac{TP}{TP+FN} \qquad \text{Precision} = \frac{TP}{TP+FP}$$

TP - truly positive signal, TN - truly negative signal,
FP - falsely positive signal, FN - falsely negative signal.



Area $= 0.5$ - completely random choice.

Area $= 1$ - completely accurate classification (impossible).

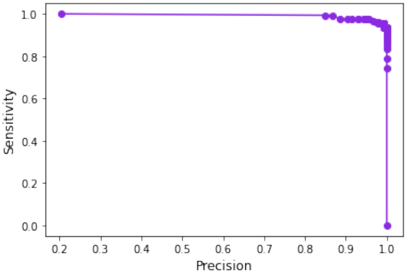
Area achieved by our classifier $=$ 0.99862 (very good).

Distinguish criteria - ROC curve cut

To separate background from signal we need to choose a **cut point** at BDT prediction axis.

Optimal cut = Best performance of the classifier

if(BDT prediction > cut) ⇒ it is **signal**
else ⇒ it is **background**



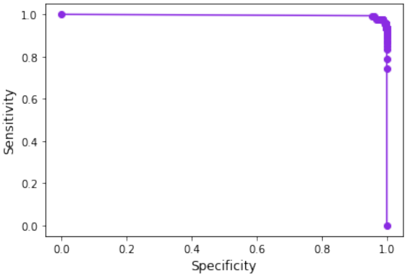
Our **optimal cut** = 0.262626...
Area achieved by our cut = 0.94776 (very good).

Other distinguish criteria - different cuts

Same rules as earlier but different way of choosing **the cut**.
Differently defined ROC curve:

1. $\uparrow$ Sensitivity.
2. $\rightarrow$ Specificity.

$$\text{Sensitivity} = \frac{TP}{TP+FN} \qquad \text{Specificity} = \frac{TN}{TN+FP}$$



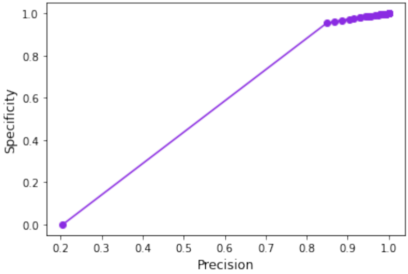
This time **optimal cut** = 0.121212...
Area achieved by this cut = 0.96236 (very good).

Other distinguish criteria - different cuts

Same rules as earlier but different way of choosing **the cut**.
Minimal distance between SP curve and $y = x$ line:

1. $\uparrow$ Specificity.
2. $\rightarrow$ Precision.

$$\text{Specificity} = \frac{TN}{TN+FP} \qquad \text{Precision} = \frac{TP}{TP+FP}$$



This time **optimal cut** = 0.131313...
Distance between SP curve and 1:1 line = 0.0001836 (very good).

What we did?

3. Application of our classifier on small data sample related to intermediate and final states.

1. data collected between 2015 and 2018,
2. prepared earlier so it can be analysed (Data sample base on run 2 sample),
3. adding classificatory variables after preselection which allow to reduce the size of input samples (into training) until we see something in mass distribution.

4. Making histograms, playing with parameters, data samples and variables, interpreting results etc.

What we did?

5*. Additional tasks.

1. Training classifier with different data set, generated using different method (multiplied).

Different data sets

The data before entering our classifier was earlier filtered and divided into several sets depending on two features.

1. Type of charge of D daughter combination - KKPi or KPiK.
2. Type of track particles leave - DD (downstream) or LL (long track).

Results

A few of plots from KKpi DD data set:

We can see a K^* meson!

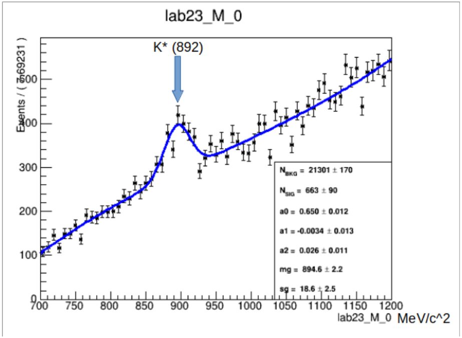


Figure: $K*$ mass distribution in MeV/c^2 - before removing the background.

Metric used to rate efficiency of the classifier:

$$\frac{S}{\sqrt{S+B}} = 4.4736$$

Results

A few of plots from KKpi DD data set:

We can see a K^* meson!

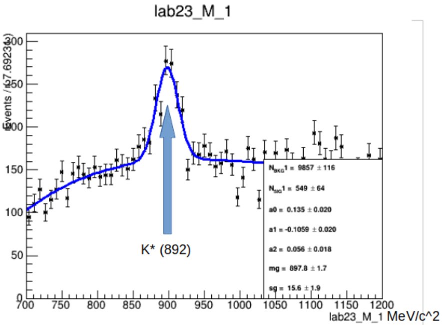


Figure: $K*$ mass distribution in MeV/c^2 - after removing the background with **cut = 0.1**.

Metric used to rate efficiency of the classifier:

$$\frac{S}{\sqrt{S+B}} = 5.3818$$

Results

A few of plots from KKpi DD data set:

We can see a K^* meson!

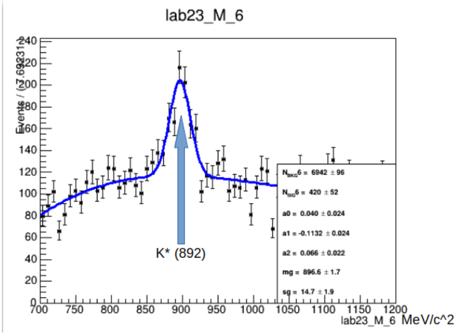


Figure: $K*$ mass distribution in MeV/c^2 - after removing the background with **cut = 0.6**.

Metric used to rate efficiency of the classifier:

$$\frac{S}{\sqrt{S+B}} = \boxed{4.8949}$$

Results

A few of plots from KKpi DD data set:

We can see a K^* meson!

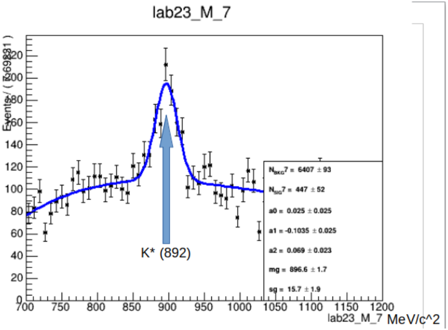


Figure: $K*$ mass distribution in MeV/c^2 - after removing the background with **cut = 0.7**.

Metric used to rate efficiency of the classifier:

$$\frac{S}{\sqrt{S+B}} = \boxed{5.3992}$$

$\frac{S}{\sqrt{S+B}}$ fluctuate because fitting algorithm is not perfect.

Conclusions

Main conclusions of this project:

1. Trained classifier performed very well.
2. This method is very quick and easy.
3. There is no one "the most optimal" way to perform such analysis - different training data sets, different values of classifier and training parameters, cut criteria etc., may give equally satisfying results.

The end

Than you for your attention.

References I

[1] A Augusto Alves Jr, LM Andrade Filho, AF Barbosa, I Bediaga, G Cernicchiaro, G Guerrer, HP Lima Jr, AA Machado, J Magnin, F Marujo, et al. The lhcb detector at the lhc. *Journal of instrumentation*, 3(08):S08005, 2008.

[2] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[3] Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, and Yuan Tang. Xgboost: extreme gradient boosting. *R package version 0.4-2*, pages 1–4, 2015.