BDT-I

Giulia Maineri

Università degli studi di Milano

March 2022

Table of Contents

TMVA Analysis

A typical TMVA Analysis consists of three main steps:

- Pre-analysis and Pre-processing
  Providing variables according to the methods invoked and apply cuts to the input variables. Correlation coefficients of the input variables and a first unspecified variable ranking are calculated.
- Training phase
  Training, testing and evaluation of classifiers using data samples with known signal and background composition. The chosen classifier computes optimal values of some tunable parameters w (weights) of the mapping function $y(\vec{x})$ in order to maximize signal-background separation
- Application phase
  Using selected trained classifiers to classify unknown data samples

Training

- Training is done only on events from Signal Region but BDT will be applied on all the events
- Variable $\Delta\Phi(E_T^{miss}, \sum \vec{p}_T^{jets})$ has been deleted as strongly correlated with $\Delta\Phi(\vec{E}_T^{miss}, \vec{p}_T^{closestjet})$ in both signal and background

Training-Factory

Factory was set with the following options:

- !V; necessary to deactivate "verbose mode", which prints explanations of what's going on
- !Silent; necessary to have a TMVA output after the creation of a factory object
- Color, in order to have a colored screen output
- DrawProgressBar
- Transformations=I;D;P;G,D List of transformations to test; formatting example: for identity, decorrelation, PCA, Uniform and Gaussianisation followed by decorrelation transformations
- AnalysisType=Classification; set the analysis type (in this case we want a discrete output variable in order to discriminate signal and background; for real variables output analysis "Regression" option should be selected)

Training-BookMethod

Trees training was done with the following options:

- !H; in order not to print a method-specific help message
- !V; necessary to deactivate "verbose mode", which prints explanations of what's going on
- NTrees=850; number of trees in the forest
- MinNodeSize=2.5%; minimum percentage of training events required in a leaf node
- MaxDepth=3; maximum depth of the decision tree allowed
- BoostType=Adaboost; boosting type
- AdaBoostBeta=0.5; learning rate for AdaBoost algorithm

Training-BookMethod

- UseBaggedBoost;
- BaggedSampleFraction=0.5; relative size of bagged event sample to original size of the data sample (used whenever bagging is used (i.e. UseBaggedGrad, Bagging,UseBaggedBoost))
- SeparationType=GiniIndex; separation criterion for node splitting
- nCuts=20;number of grid points in variable range used in finding optimal cut in node splitting

AdaBoost

The idea behind Adaptive Boosting is that for the signal events from the training sample that end up in a background node and vice versa are given a **larger weight** than events that are in the correct leaf node. The weights of previously misclassified events are multiplied by common boost weight α; the boost weight is derived from the misclassification rate E of the previous tree $\alpha = (1 - E)/E$. The weights of the entire event sample are then re-normalized such that the sum of weights remains constant. In this way future trees learn to deal with tree with a higher weight (which have been misclassified) better.

This results in a re-weighed training event sample, with which then a new decision tree can be developed. The boosting can be applied several times (typically 100-150 times) and one ends up with a set of decision trees (a forest).

AdaBoost

A single tree is not very strong. We need a Random Forest, which is an ensemble of different trees. The sum of weak learners results in a stronger learner, even if their separation power is the same if uncorrelated. The obtained forest is a single classifier, whose output is a weighted average of the individual DT.

$$y(\vec{x}) = \frac{1}{N_{collection}} \sum_{i}^{N_{collection}} \ln(\alpha_i) h_i(\vec{x})$$

where $h_i(\vec{x})$ is a single tree respones, $h_i(\vec{x}) = +1$ for signal and $h_i(\vec{x}) = -1$ for background.

AdaBoostBeta

The learning rate of the AdaBoost algorithm is controlled by a parameter β given as an exponent to the boost weight $\alpha \longrightarrow \alpha^{\beta}$.
The algorithm's robustness can be enhanced by reducing the learning rate. A small shrinkage (0.1–0.3) demands more trees to be grown but can significantly improve the accuracy of the prediction in difficult settings.
Learning rate defines how much weights are changed at each step.
If learning rate is set too low, training will progress very slowly as very tiny updates to the weights are made. If learning rate is set too high, it can cause undesirable divergent behavior in the algorithm's behaviour.

UseBaggedBoost

A classifier (i.e. the whole Forest) is repeatedly trained using resampled training events such that the combined classifier represents an average of the individual classifiers and is more stable with respect to statistical fluctuations in the training sample.

Resampling includes the possibility of replacement, which means that the same event is allowed to be (randomly) picked several times from the parent sample. This is equivalent to regarding the training sample as being a representation of the probability density distribution of the parent sample. Resampling can be implemented by selecting only a subset of the parent sample, which can be set using *BaggedSampleFraction* option. Another resampling method consists in choosing a subset of variable for the following Forest training, which can be done using *UseNVars* option.

Gini Index

Gini Index is a quantity used to compute a node's gain. Separation gain is computed subtracting to the parent cell's gain the two daughter's cells one. The aim is to maximize separation gain.

Gini Index is defined as

$$p(1-p)$$

where $p = \frac{S}{S+B}$ is purity.

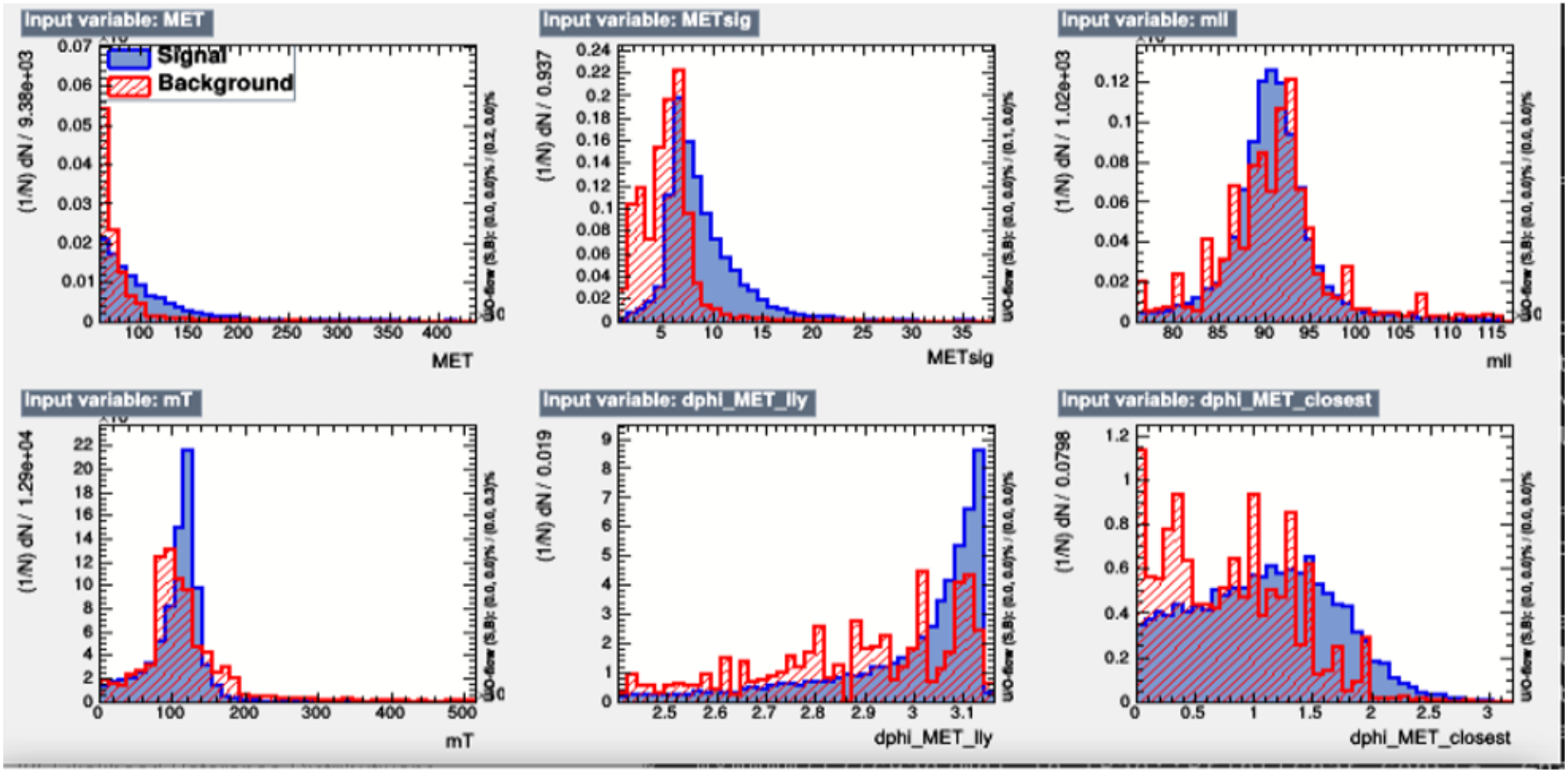
Training-BookMethod

nCuts
In order to maximize separation gain, the training procedure selects the optimal variable and cut value.
Optimal variable for the first split choice is based on the first variable ranking; following splits are done depending on others variable rankings done on the new data subests.
The cut values are optimised by scanning over the variable range with a granularity that is set via the option *nCuts*.

A truly optimal cut is determined by setting $nCuts = -1$. This invokes an algorithm that tests all possible cuts on the training sample and finds the best one, but it's much slower.

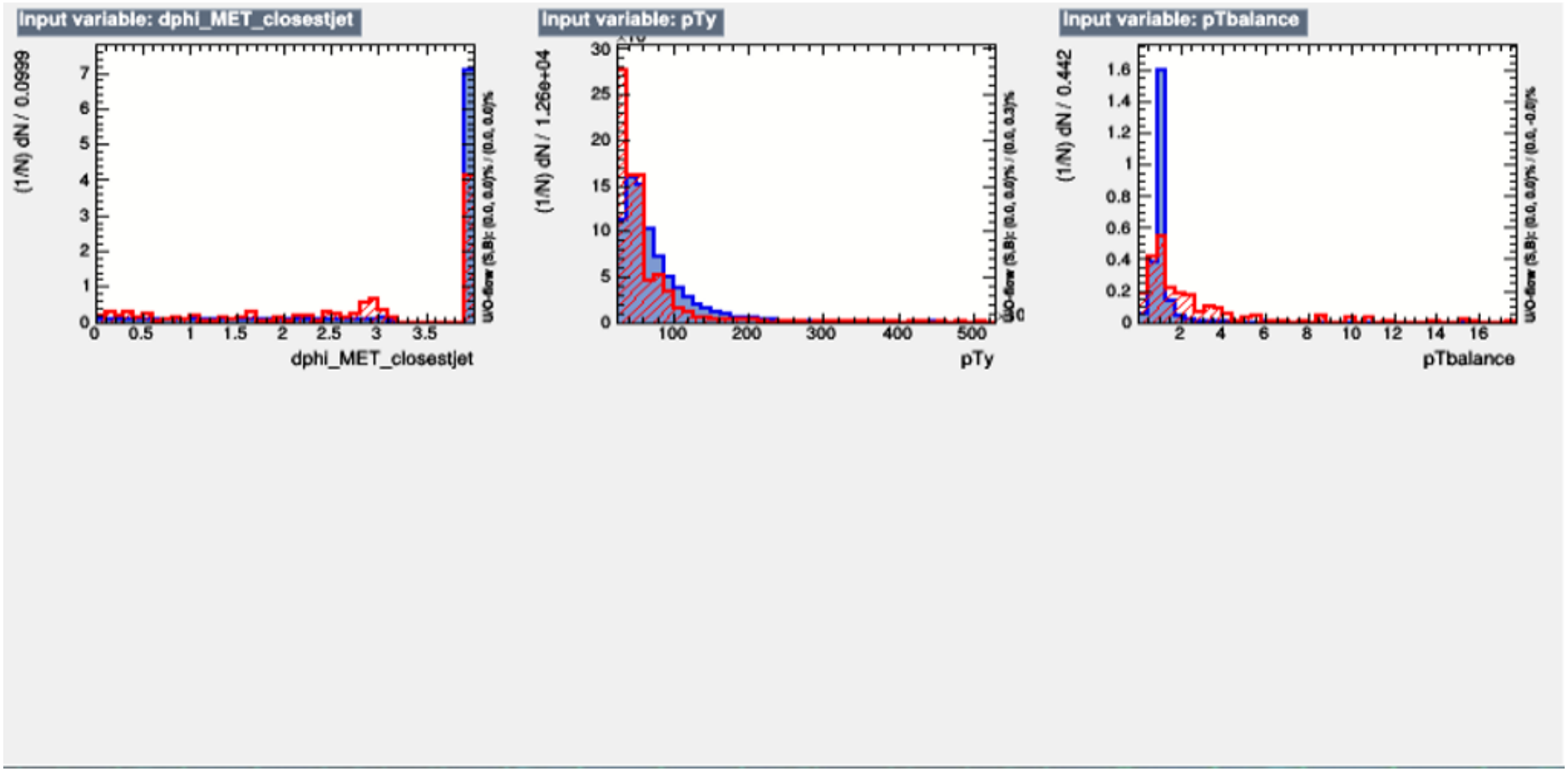
Input variables

Figure 1: MET, MET significance, Leptons invariant mass, Transverse mass, MET-llγ angle, MET-closest object angle -BDT-00



Input variables

Figure 2: MET-closest jet angle, Photon transverse momentum, Transverse momentum balance -BDT-00



Input variables

Notes

- MET significance seems to be the most discriminating variable
- $\Delta\Phi(\vec{E}_T^{miss}, \vec{p}_T^{\gamma ll})$ also seems discriminating, as it is peaked around π for signal while it has a flat distribution for background

Linear correlation coefficients

Figure 3: Signal-BDT00

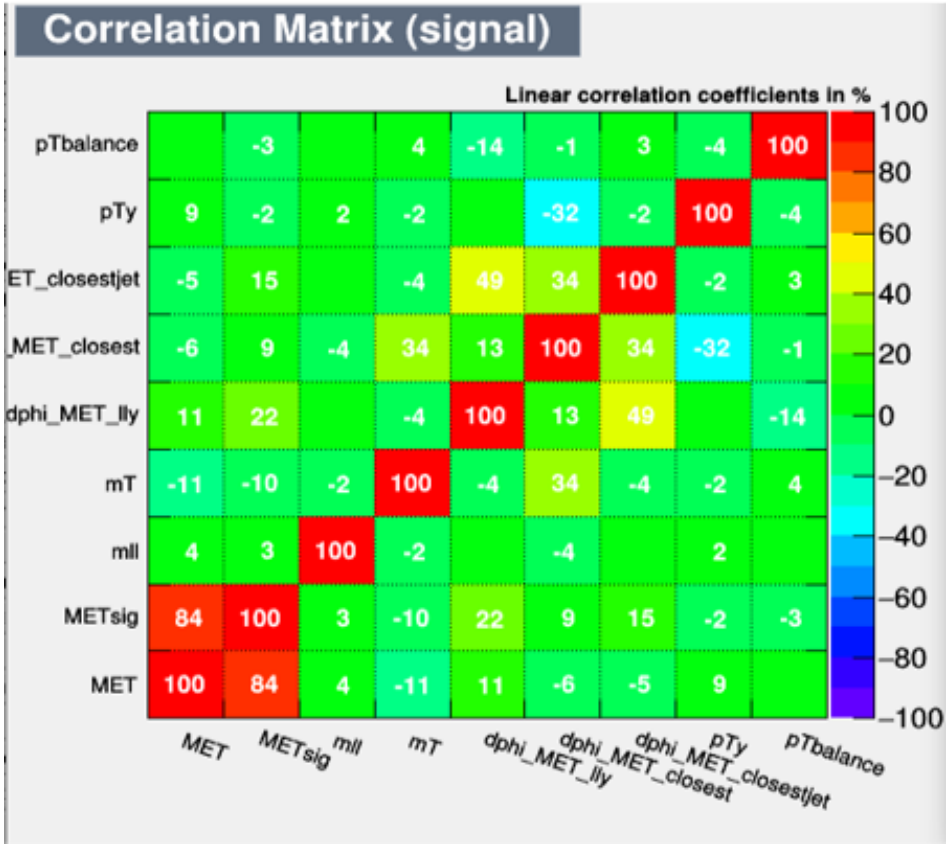
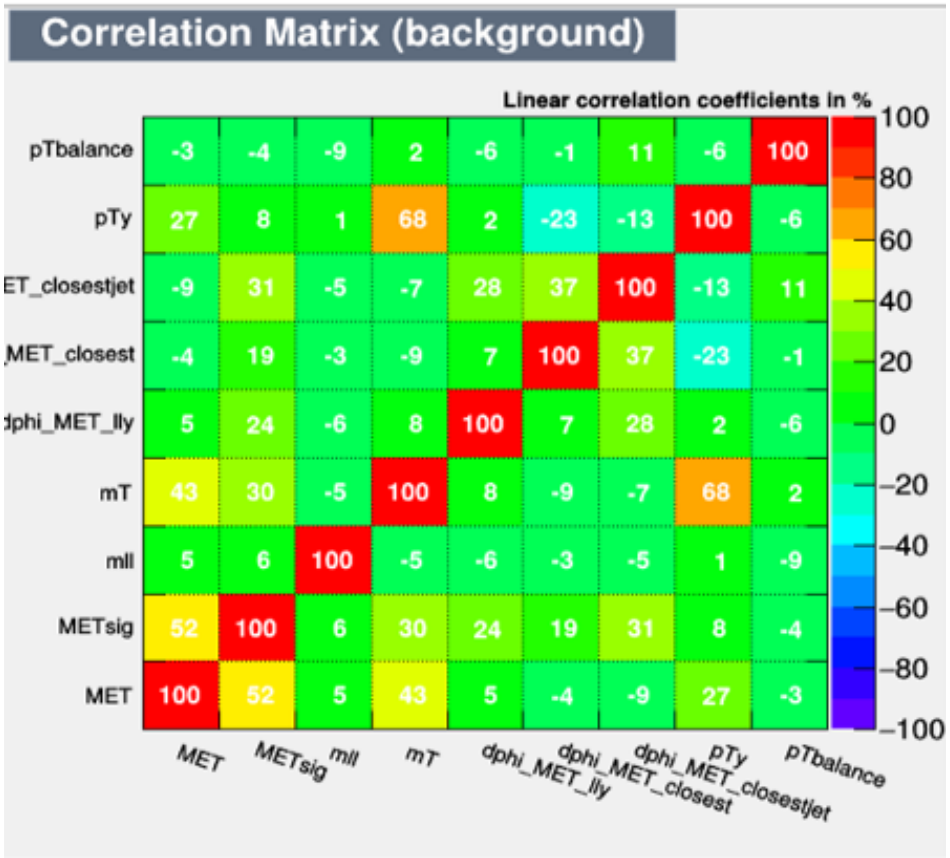


Figure 4: Background-BDT00



Linear correlation coefficients

Figure 5: Signal-BDT01

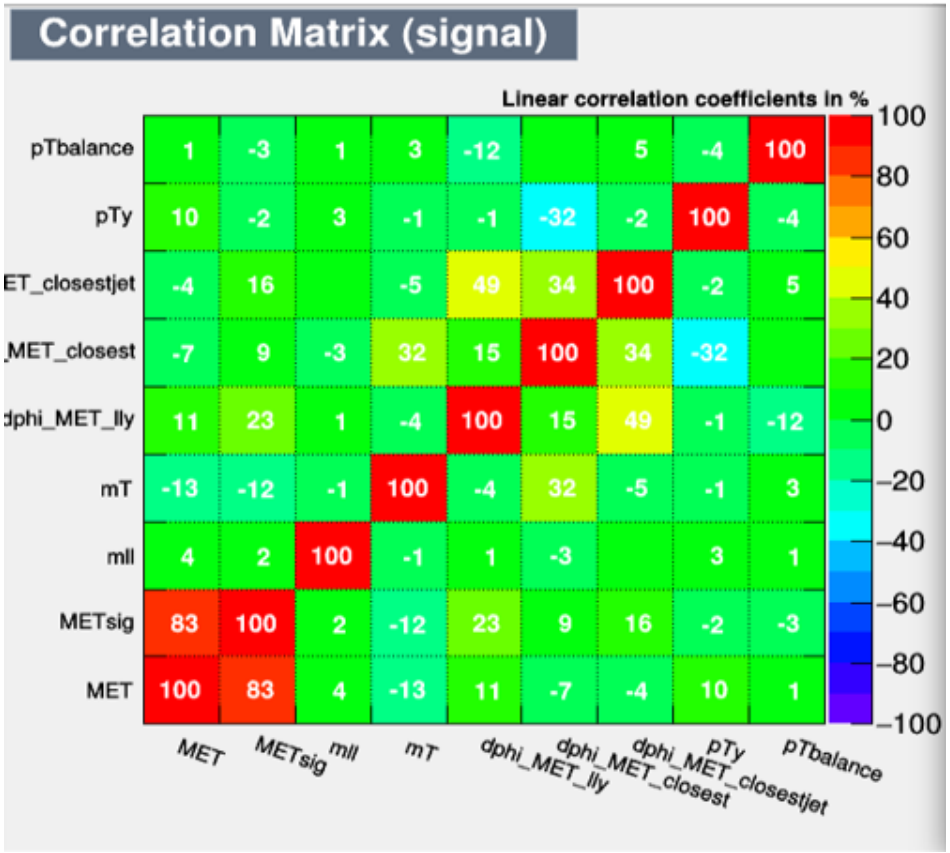
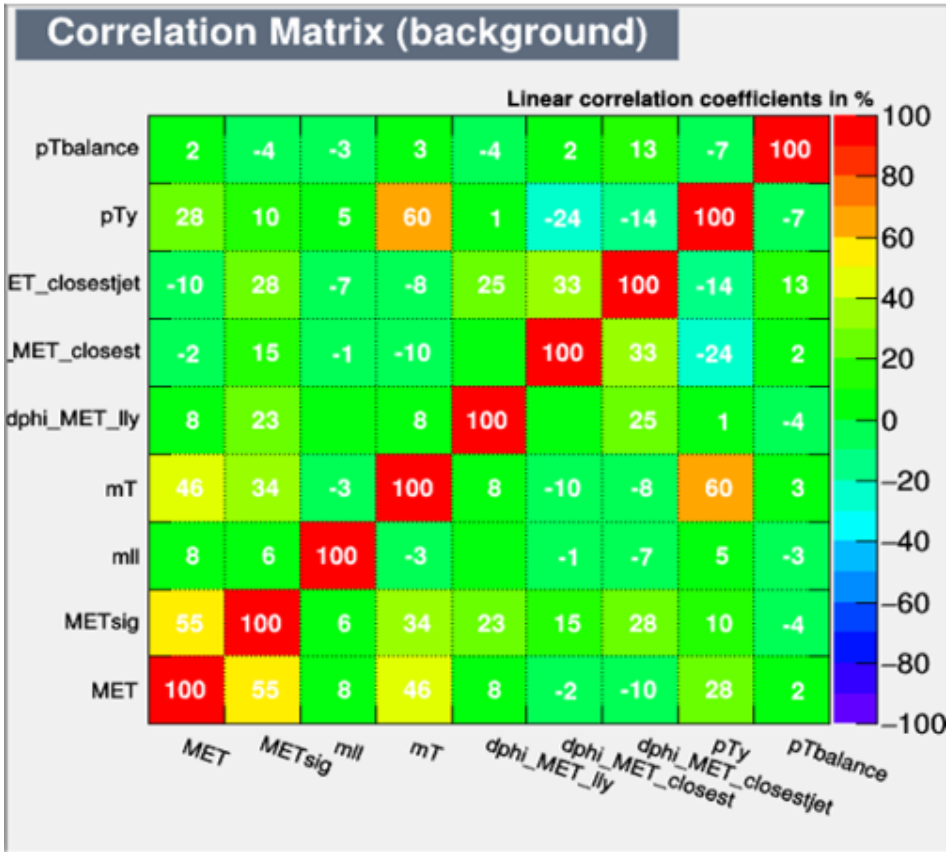


Figure 6: Background-BDT01



Linear correlation coefficients

Notes

- MET and MET significance are much more related for signal
- In background there is high correlation between p_T^{γ} and m_T, E_T^{miss} and m_T

Classifier output distribution-Test sample

Figure 7: BDT00

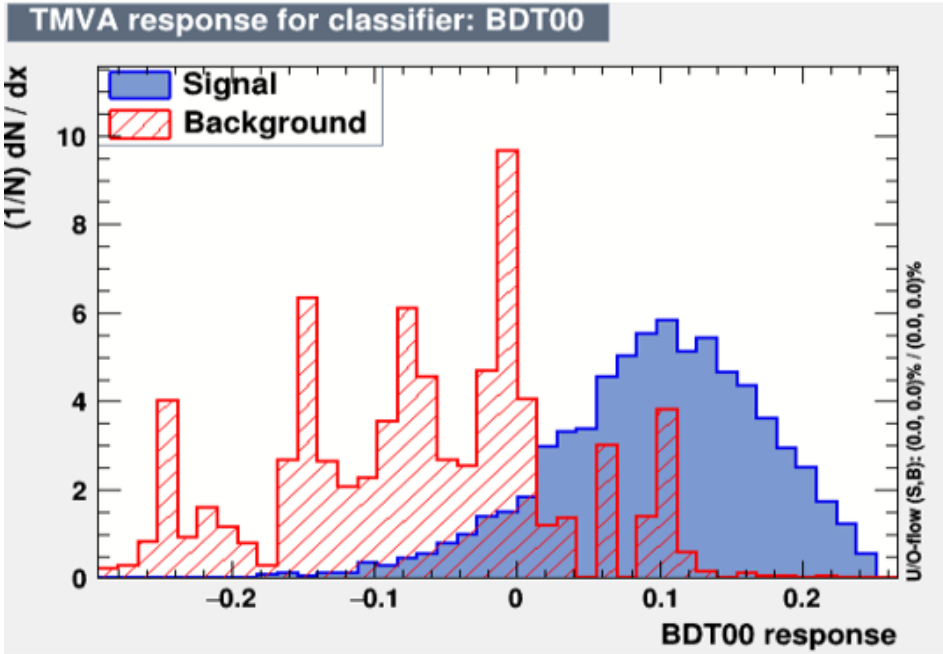
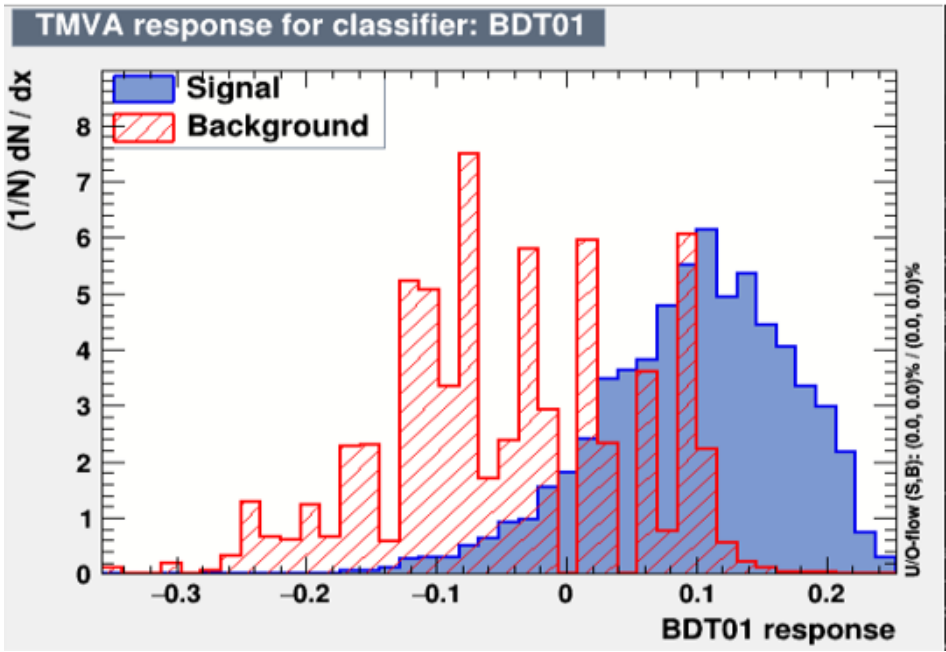


Figure 8: BDT01



Classifier output distribution-Test and training sample

In order to do an over-training control

Figure 9: BDT00

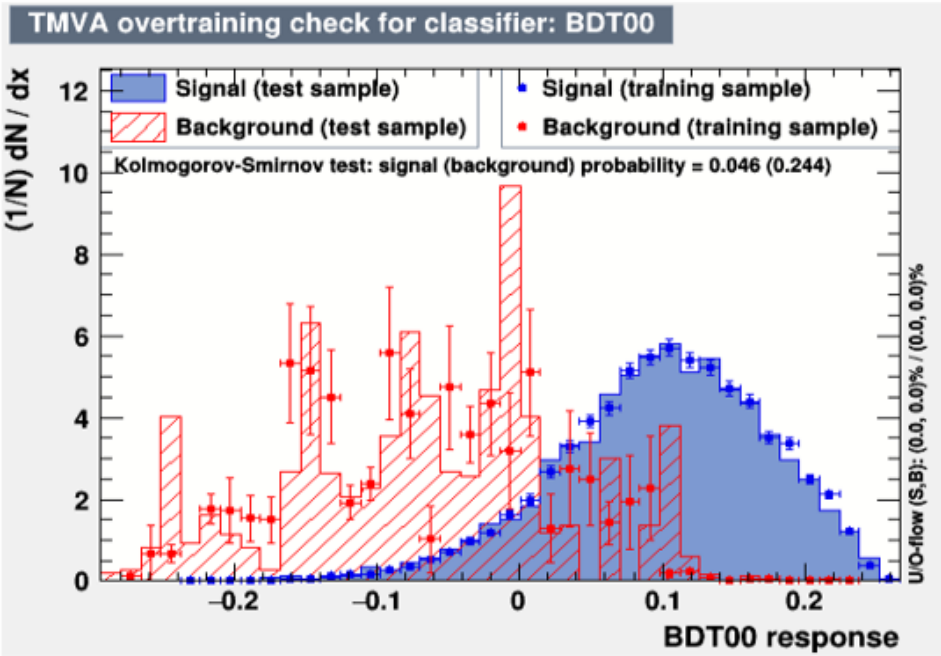
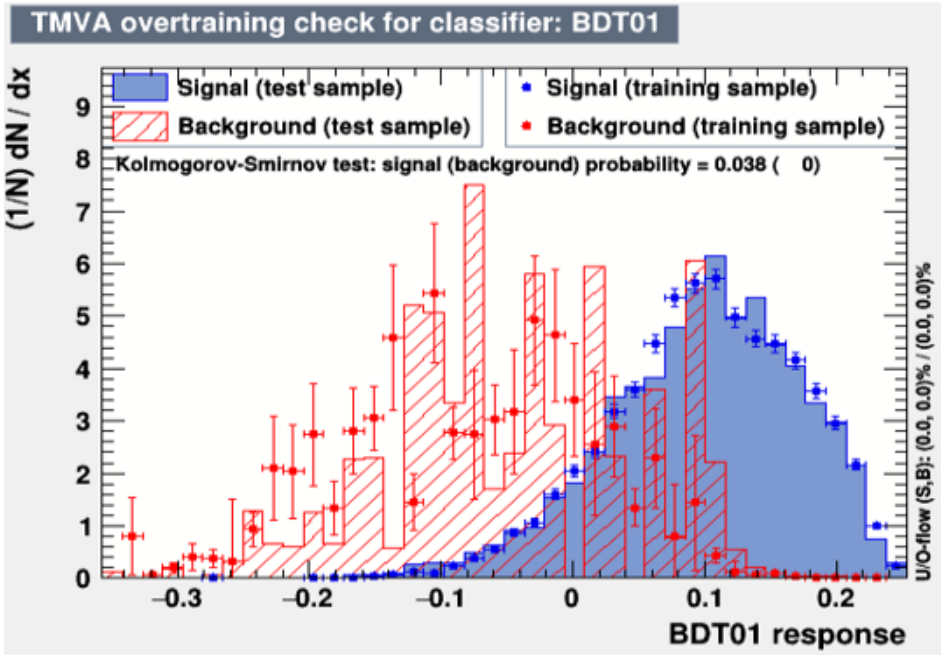


Figure 10: BDT01



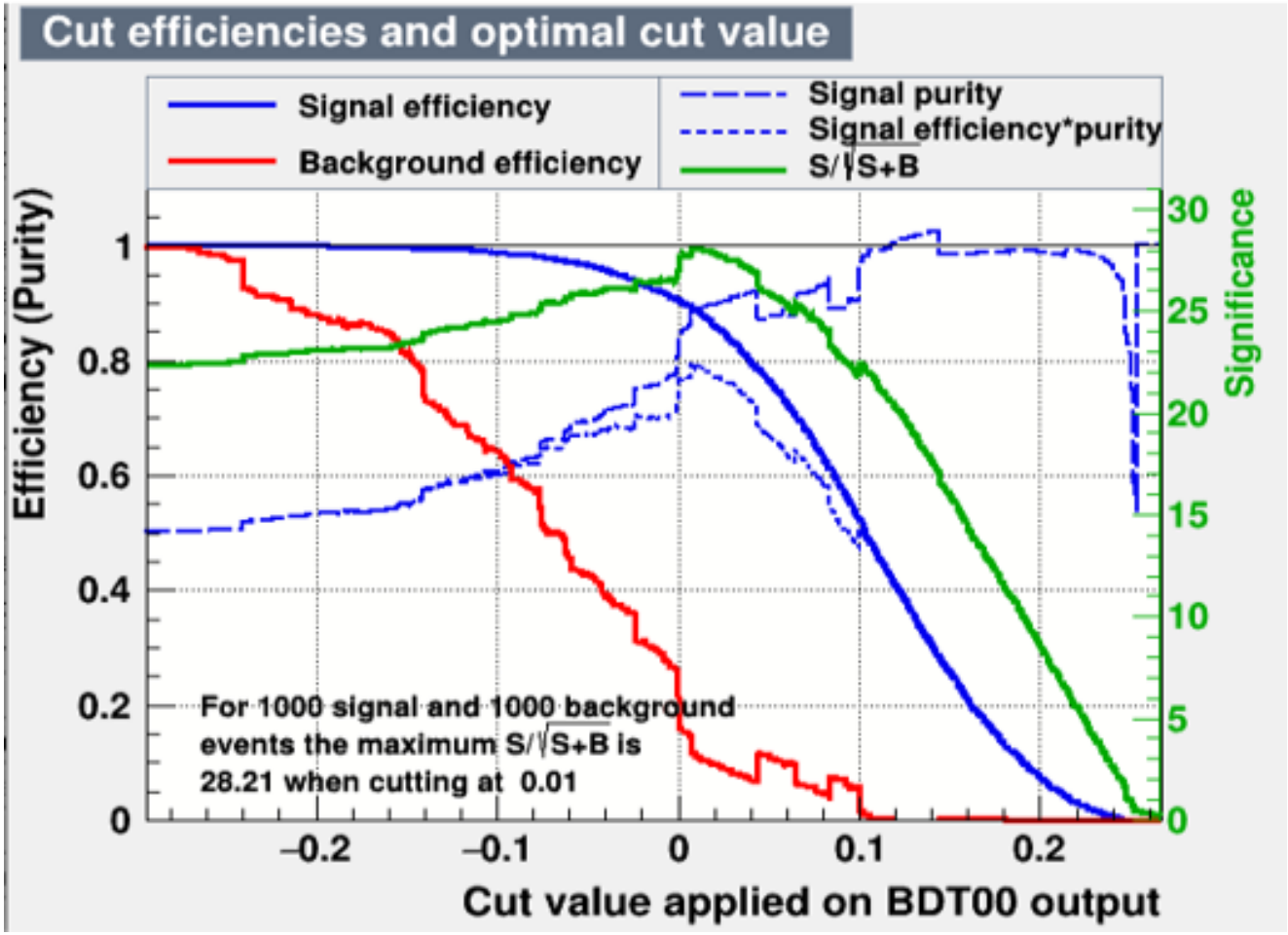
Classifier output distribution

Notes

- Training and testing samples give similar BDTs responses: this means that no over-training effect occurred
- Training and testing samples have to be different in order to avoid that results strongly depend on the training sample, thus leading to a performance deterioration when it is measured on an independent sample

Cut efficiencies

Figure 11: Cut efficiencies



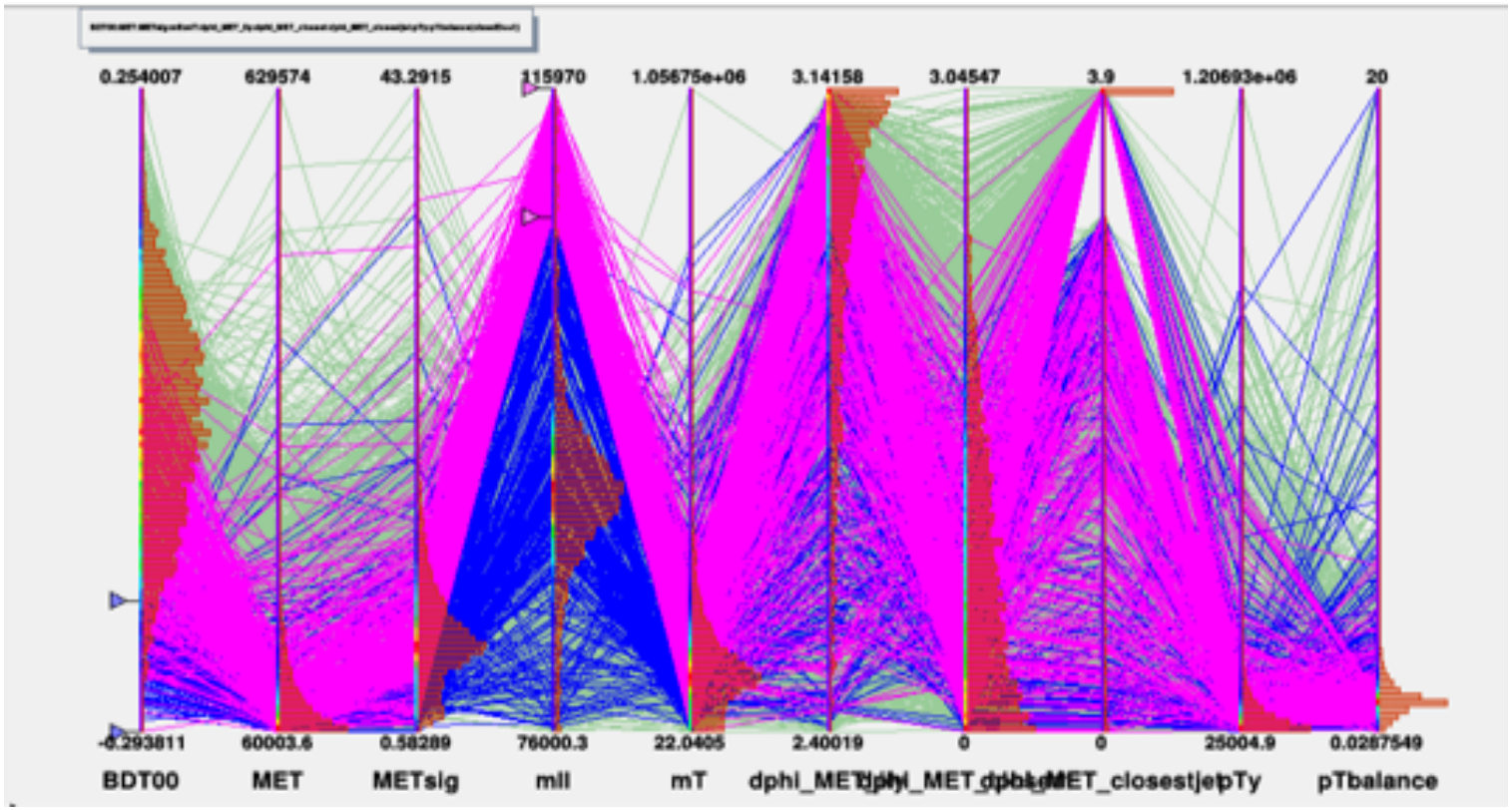
Cut efficiencies

In order to understand if signal S is present in significant quantity in the total number of events $N = S + B$, the variable $\frac{S}{\sqrt{(S+B)}}$ where at denominator there is the poissonian fluctuations of all the events. Figures of merit are built in order to understand how good the method is. A cut on BDT score makes both signal and background efficiency decrease, but the latter should decrease more rapidly. An optimal cut in BDT score can be found in order to maximize the ratio $\frac{S}{\sqrt{(S+B)}}$.

Parallel coordinate representation for BDT00 and input variables

Lines represent the paths of each event

Figure 12: Correlations among the input variables, and among the classifier and the input variables



ROC curves

ROC Curves (Receiver Operating Characteristic) are a good way to illustrate the performance of given classifier. They show the background rejection over the signal efficiency of the remaining sample. The best classifier can be identified by the largest AUC.

ROC curves

Figure 13: BDT00

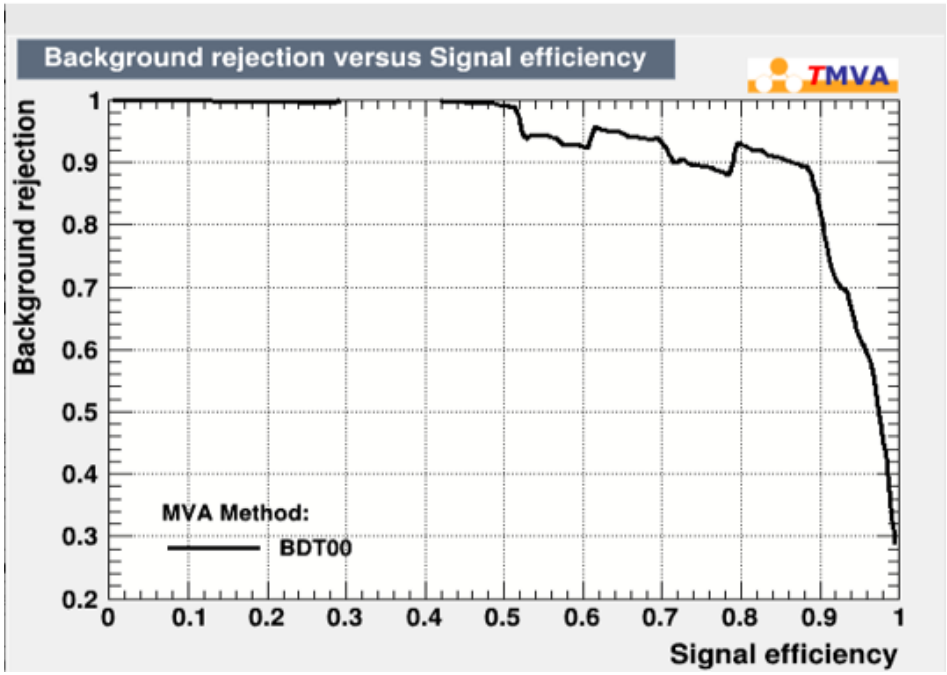
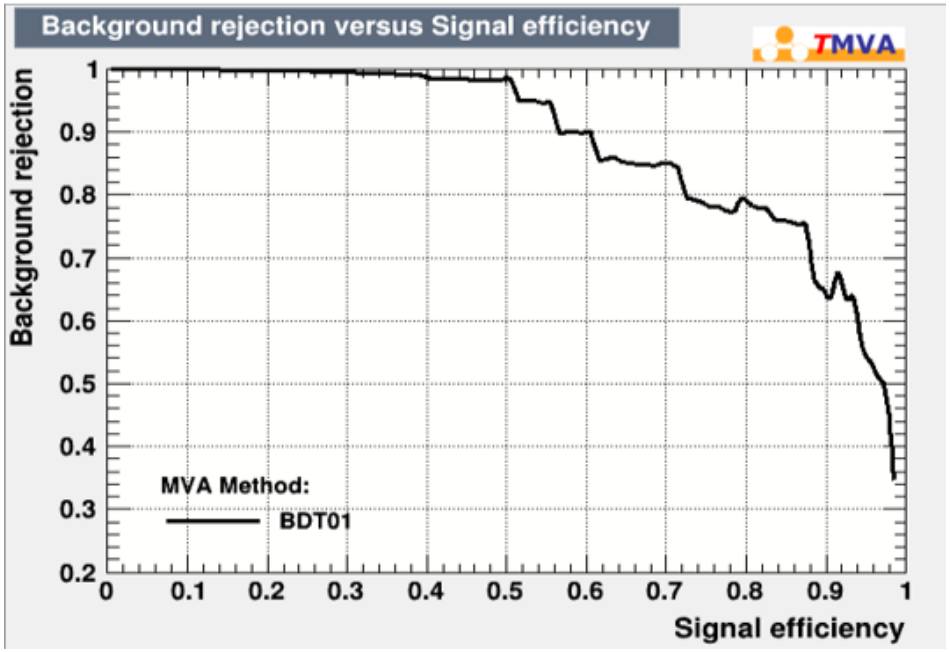


Figure 14: BT01



ROC curves

Figure 15: BDT00

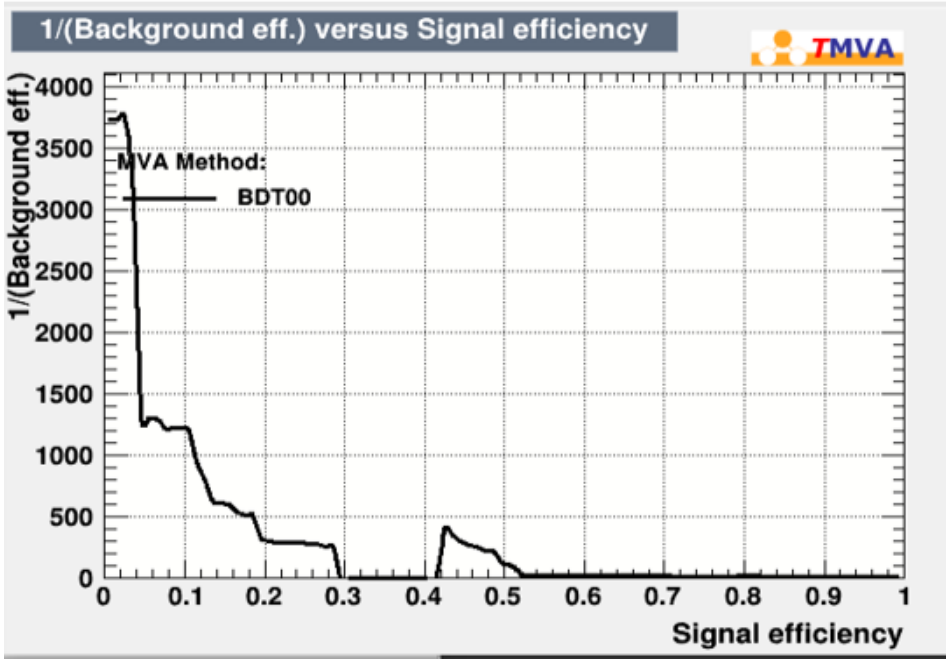
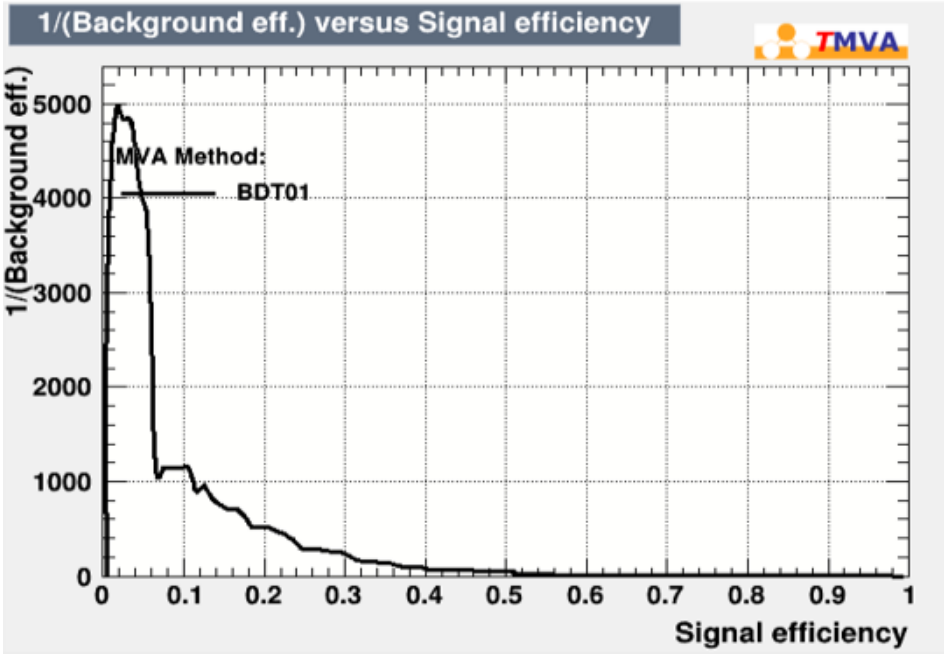
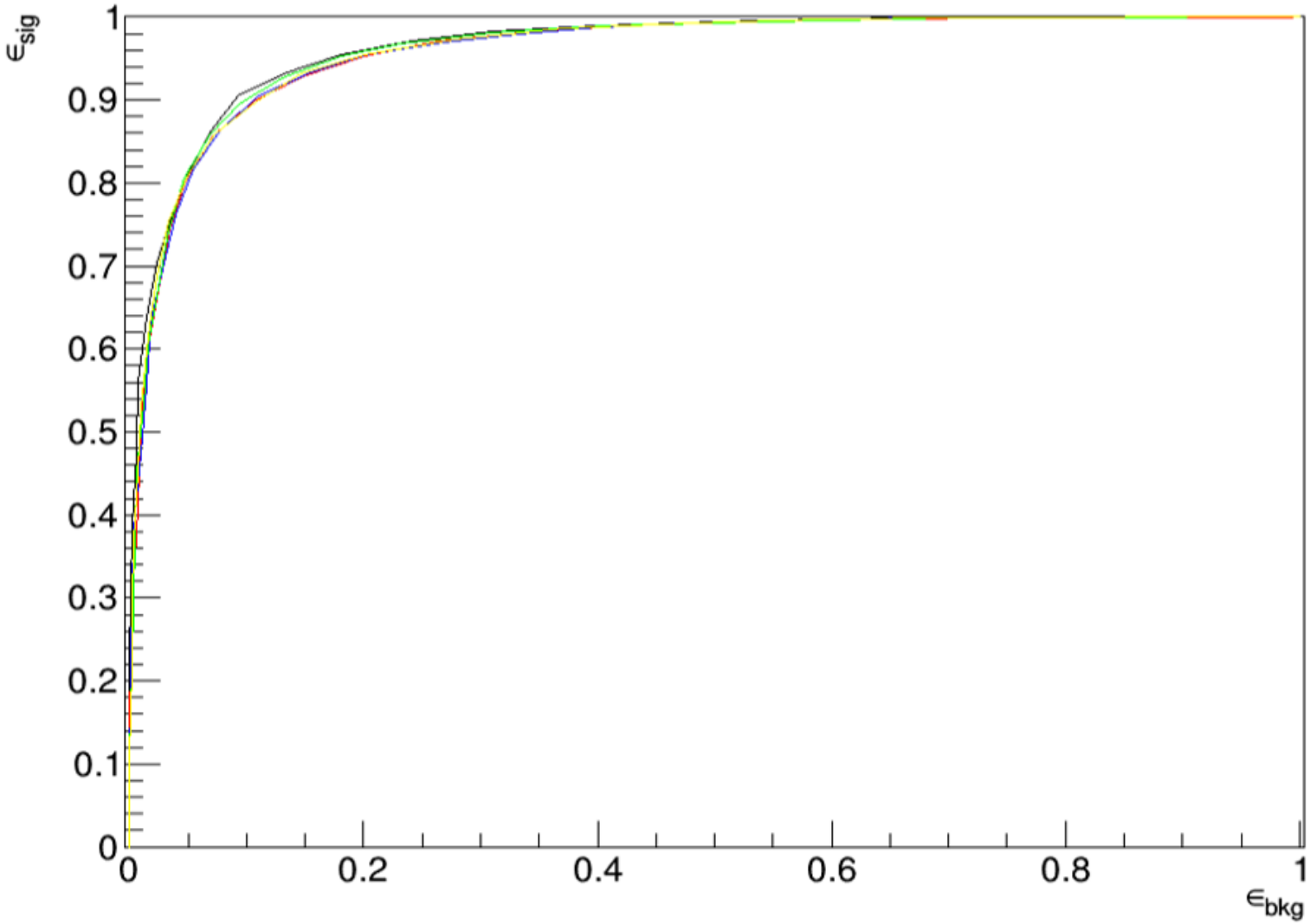


Figure 16: BT01



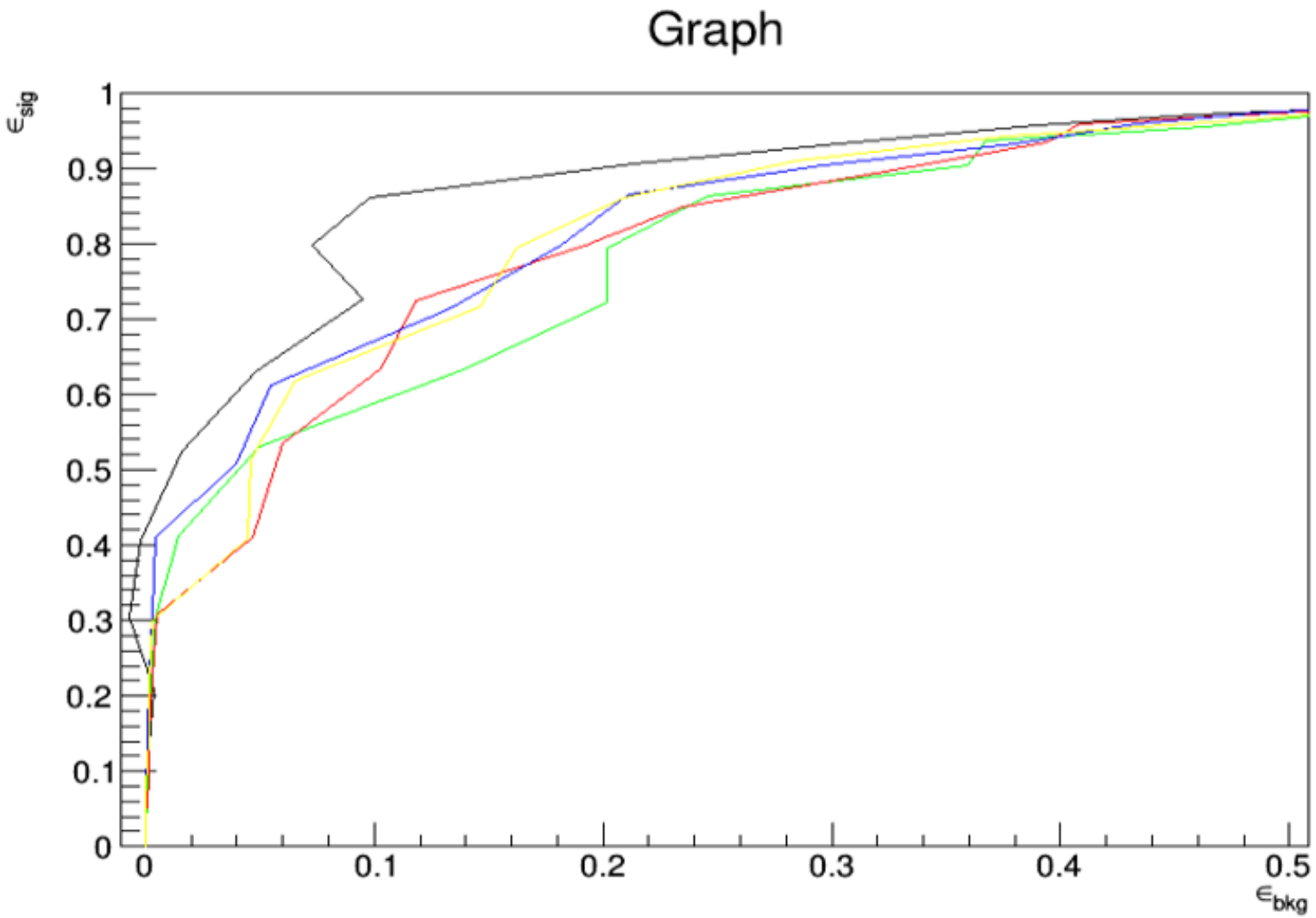
ROC curves - All training samples

Figure 17: ROC fold



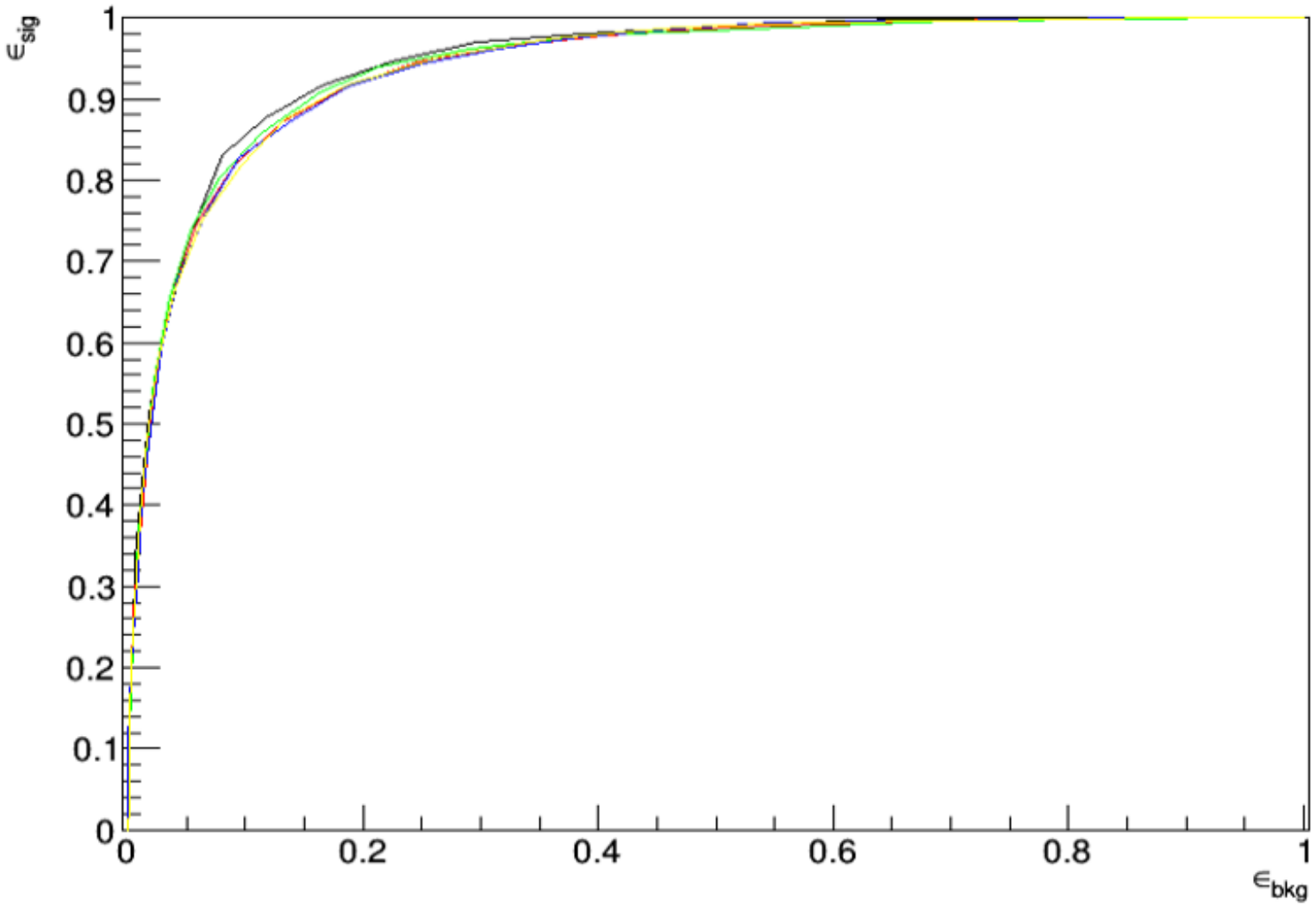
ROC curves - All training samples

Figure 18: ROC fold in Signal Region



ROC curves - All training samples

Figure 19: ROC fold not in Signal Region

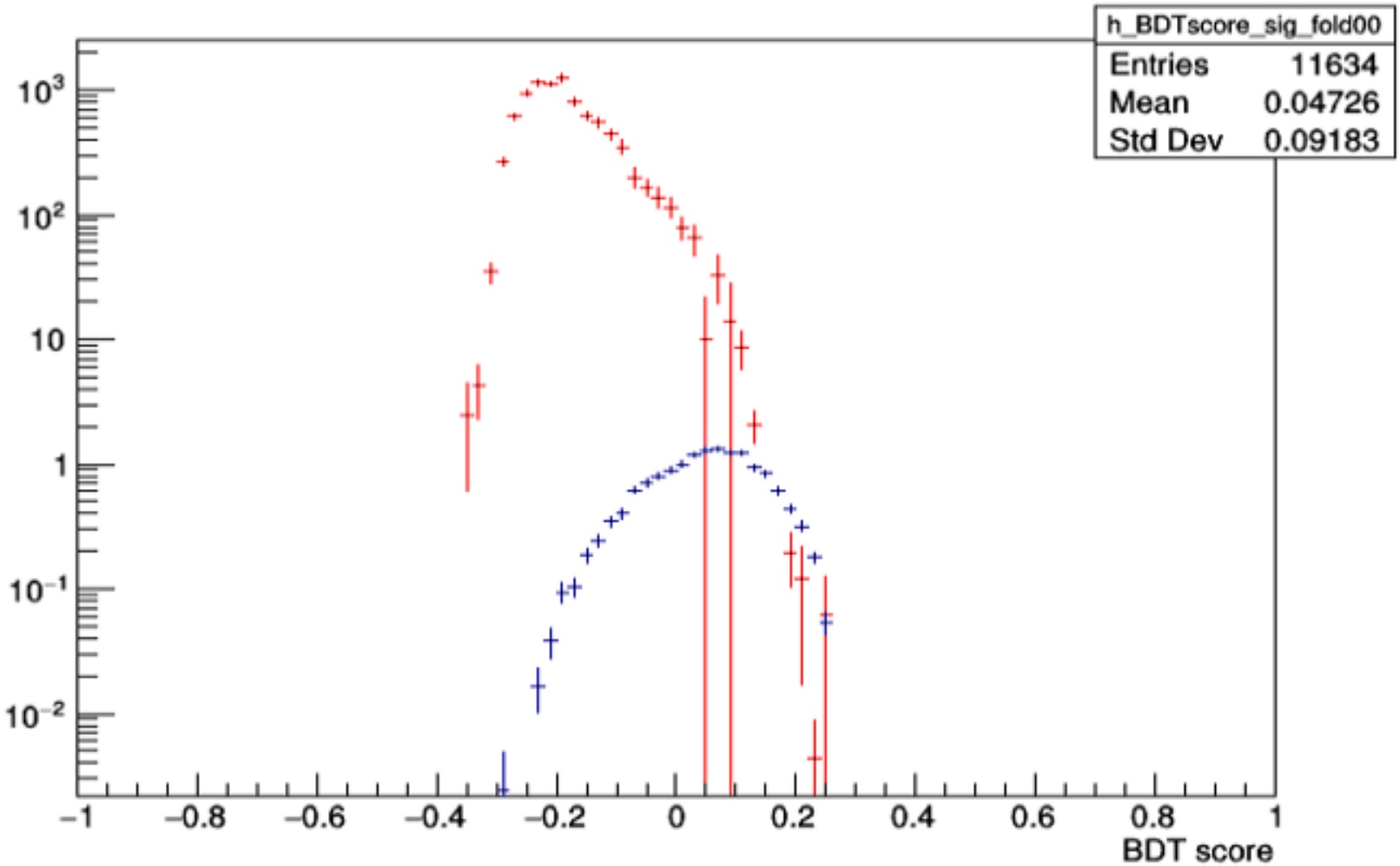


ROC curves - All training samples

ROC curve in Signal Region is much less smooth due to negative weights of background events, which can be assigned if generators are NLO (next-to-leading order). These MonteCarlo generators contain 1-loop perturbative calculations and assign negative or positive weigths depending on each event's cinematic.

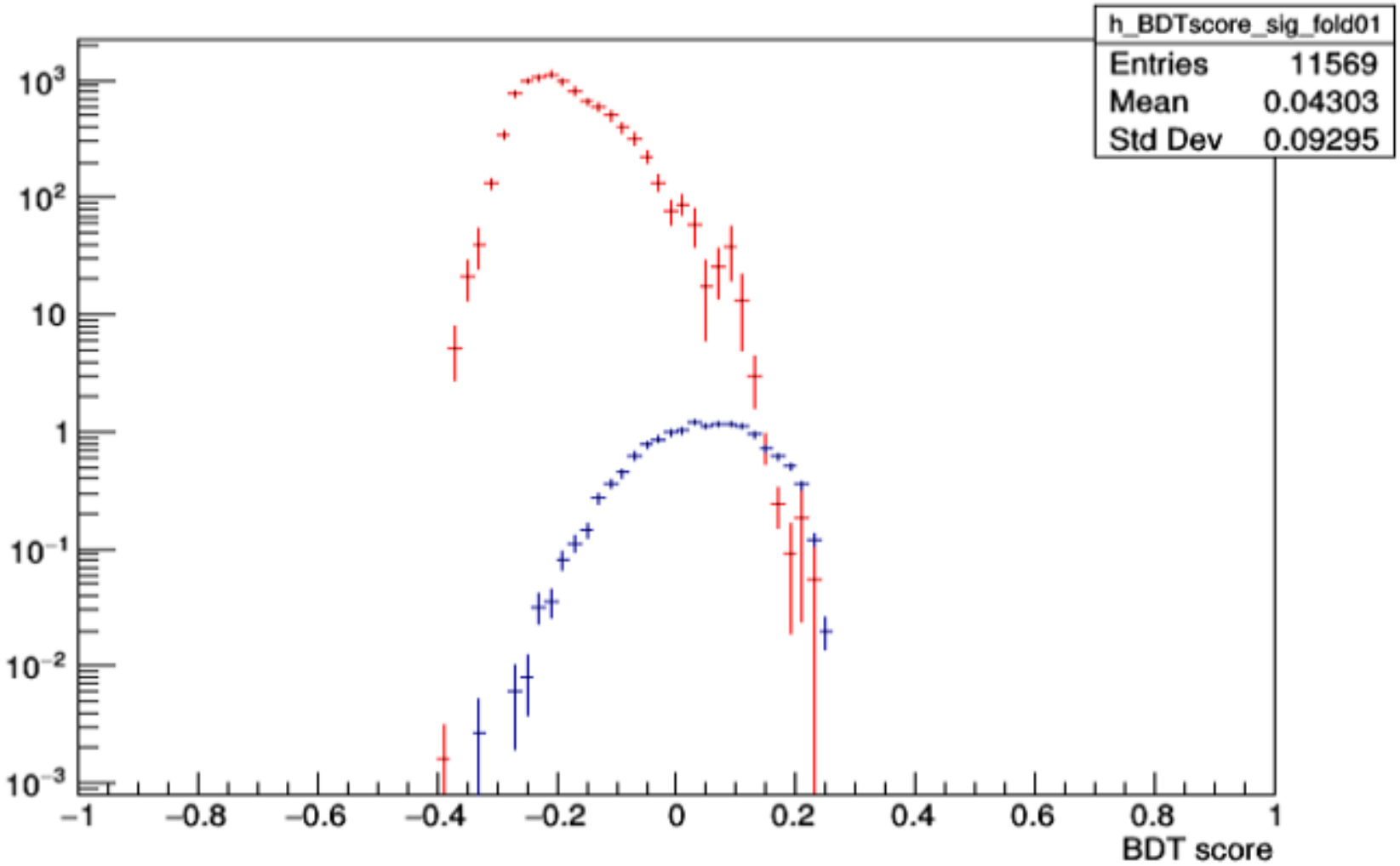
BDT score 00

Figure 20: BDT score



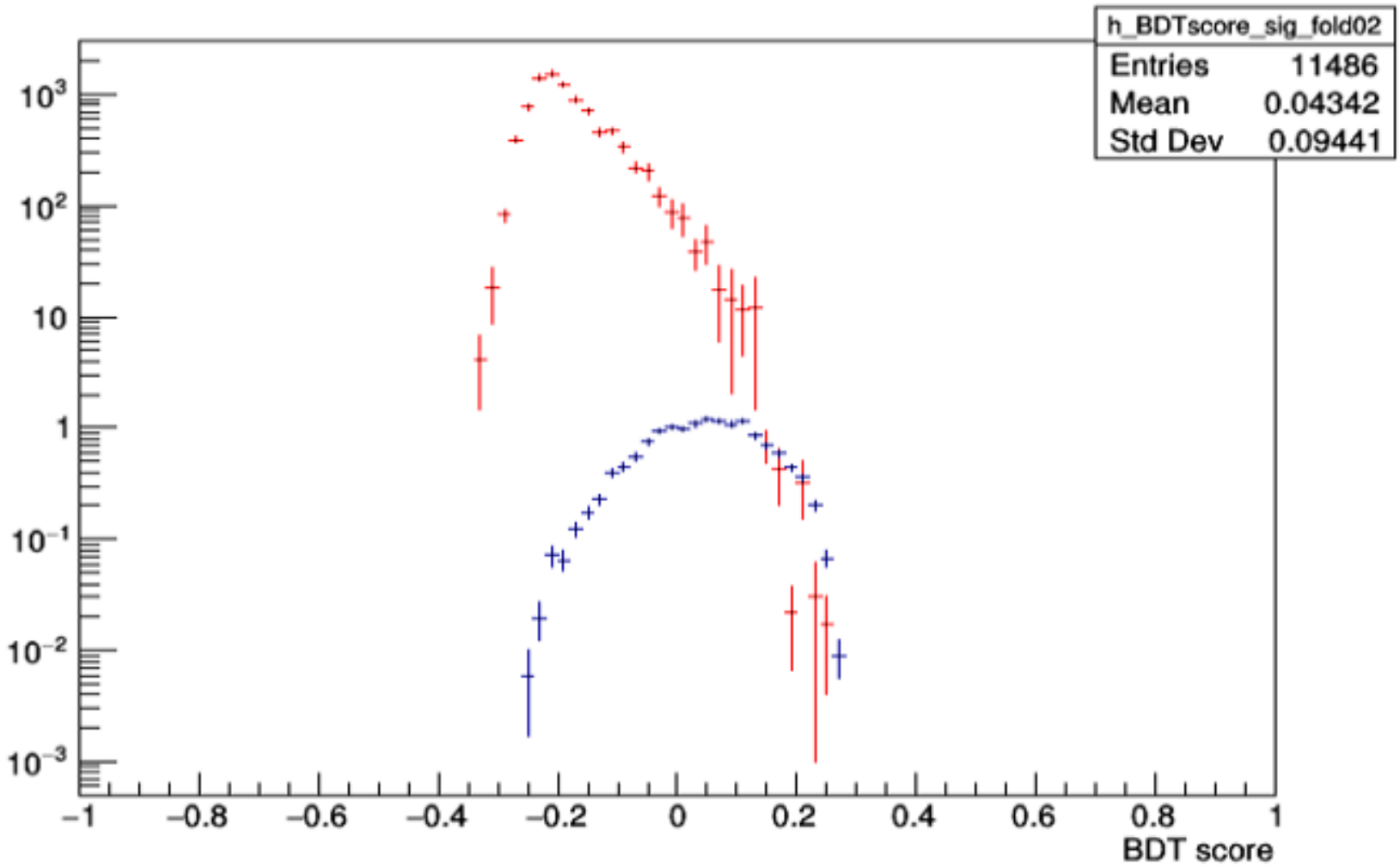
BDT score 01

Figure 21: BDT score



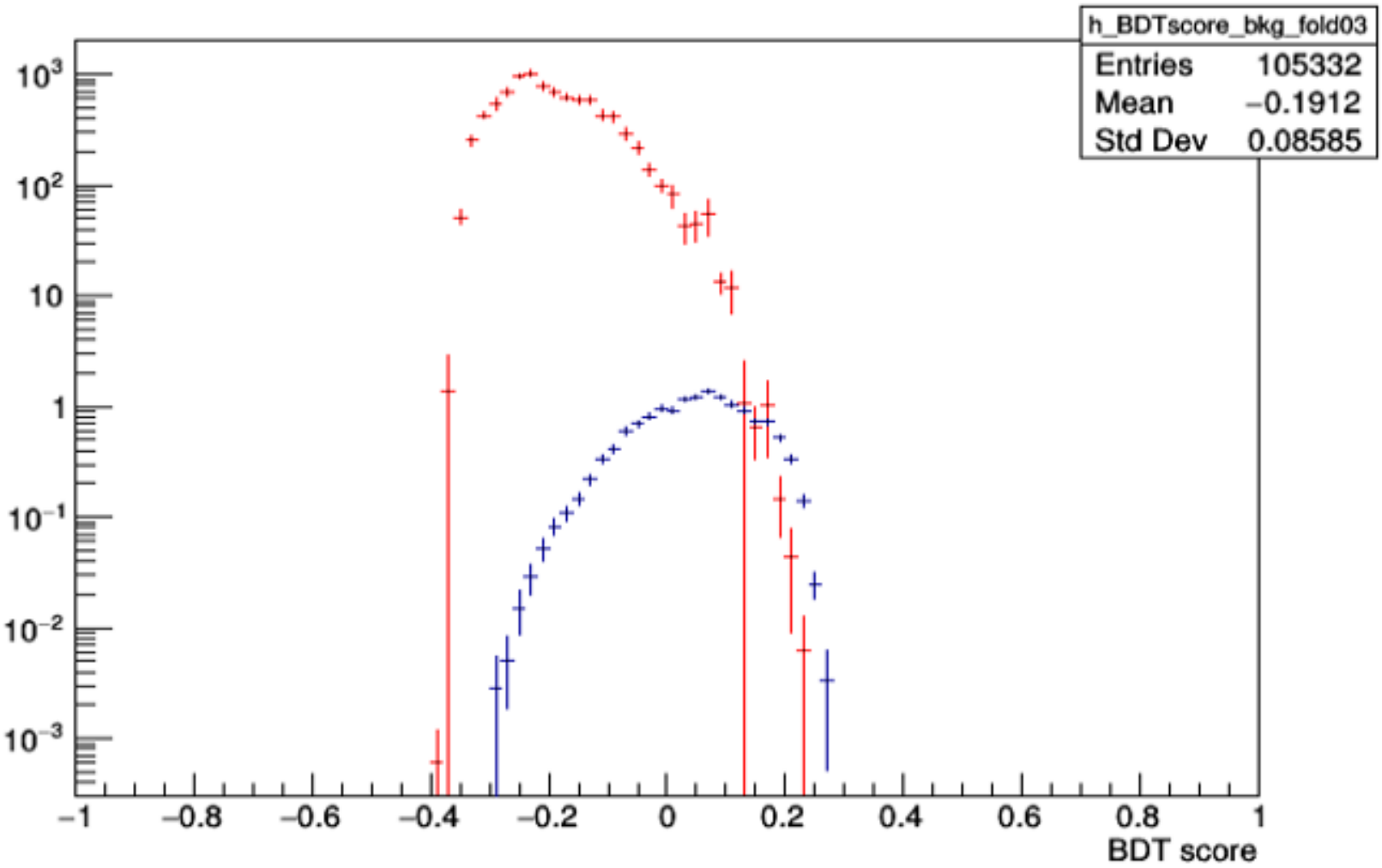
BDT score 02

Figure 22: BDT score



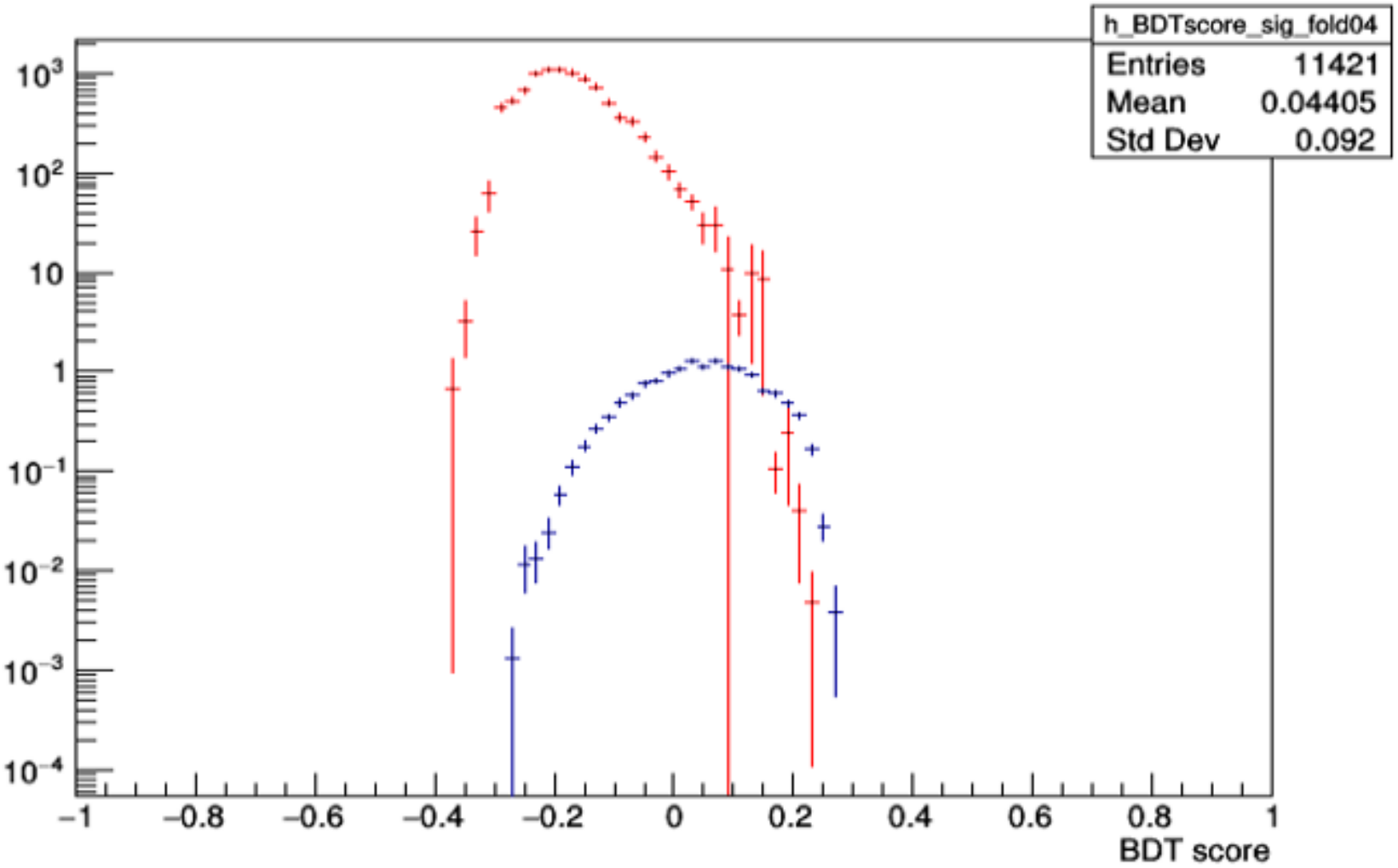
BDT score 03

Figure 23: BDT score



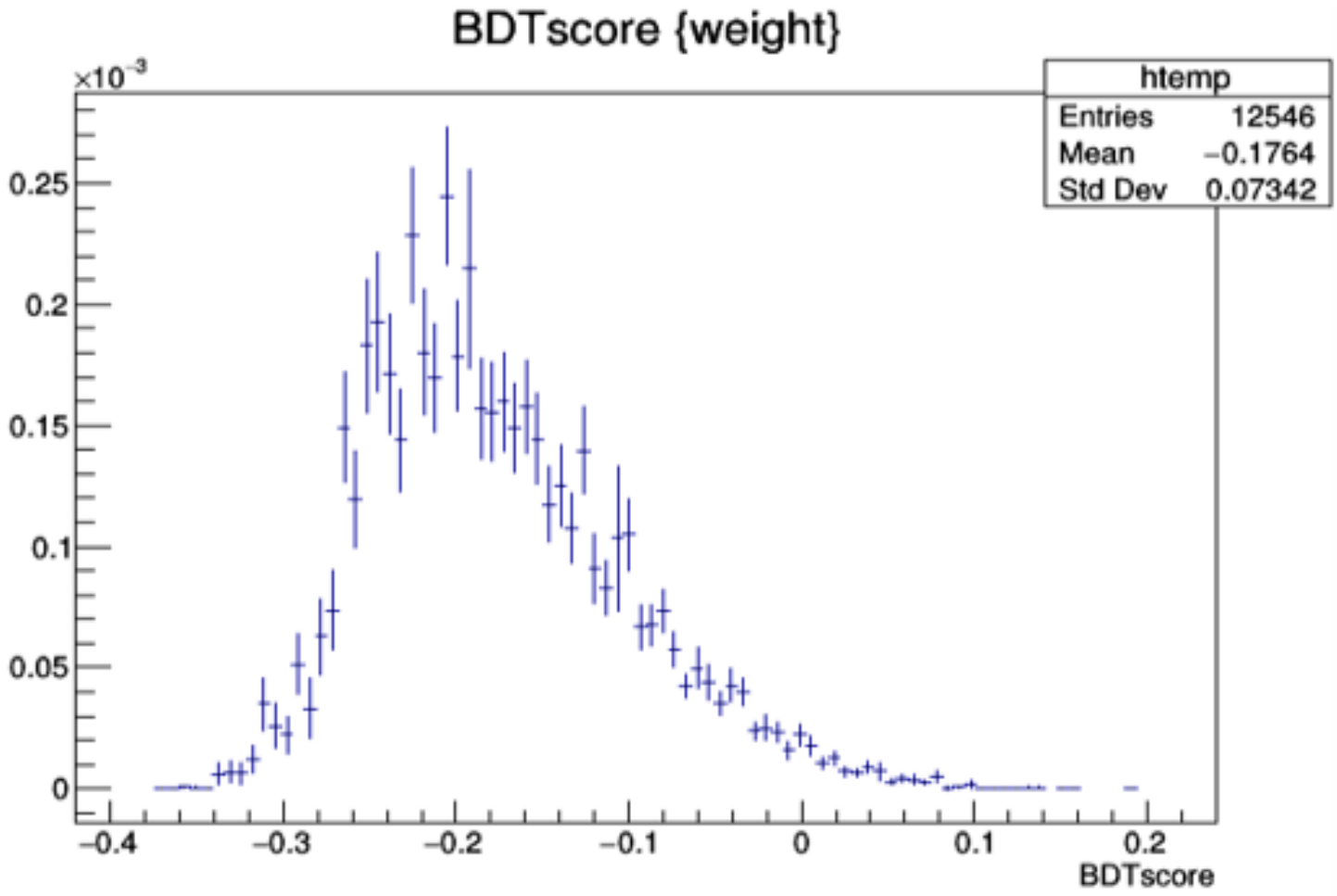
BDT score 04

Figure 24: BDT score



BDT score comparison - background

Figure 25: BDT score - V$\gamma\gamma$



BDT score comparison - background

Figure 26: Zγ strong

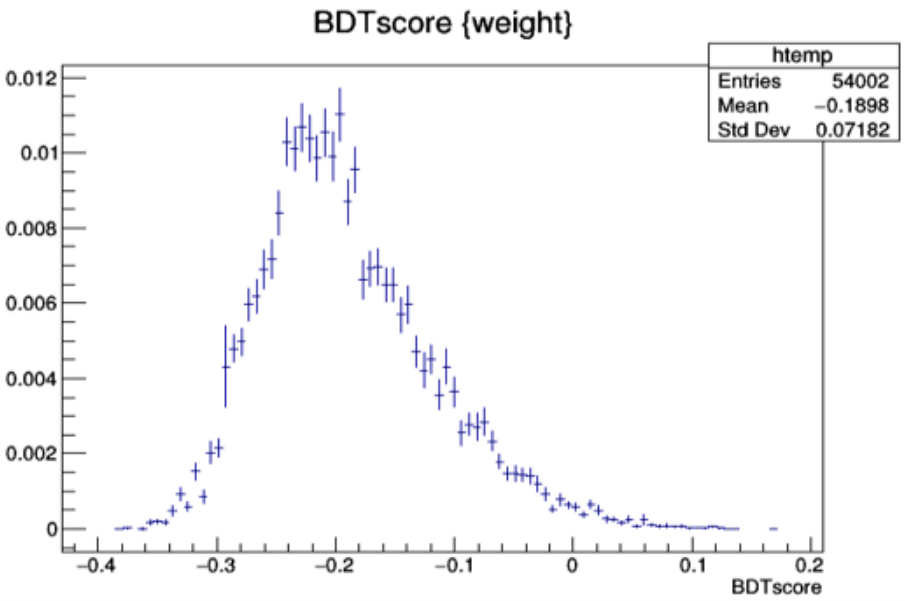
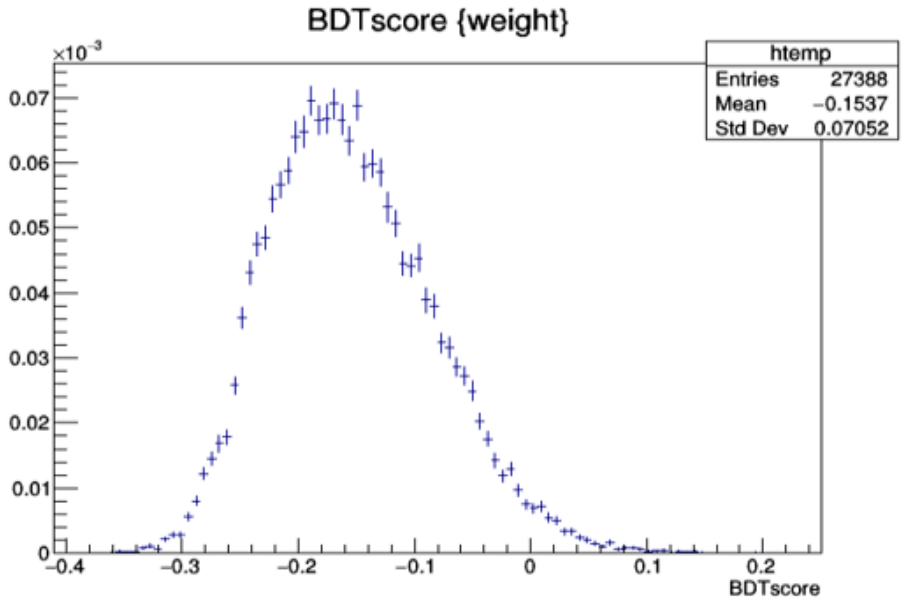


Figure 27: Zγ eweak



BDT score comparison - background

Figure 28: Z strong

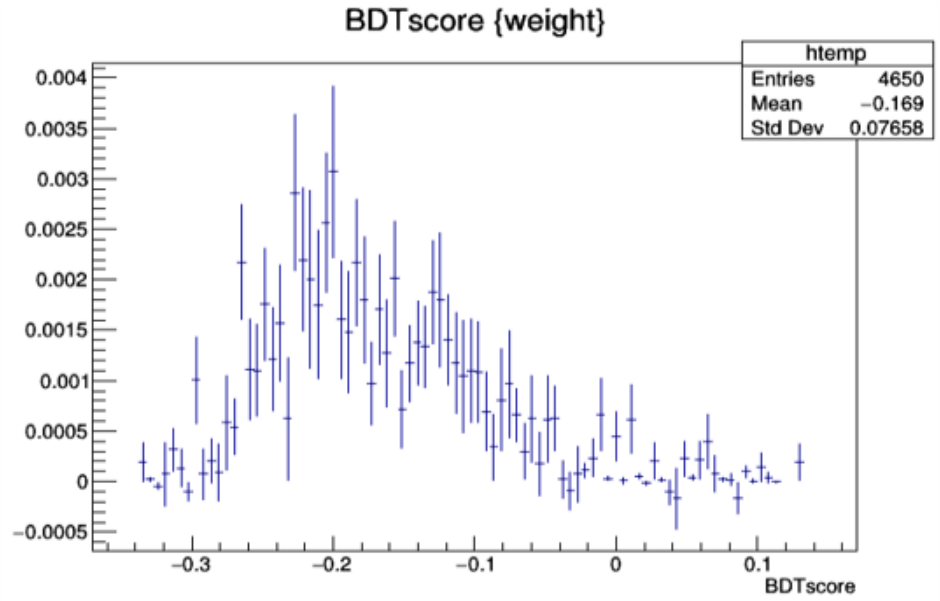
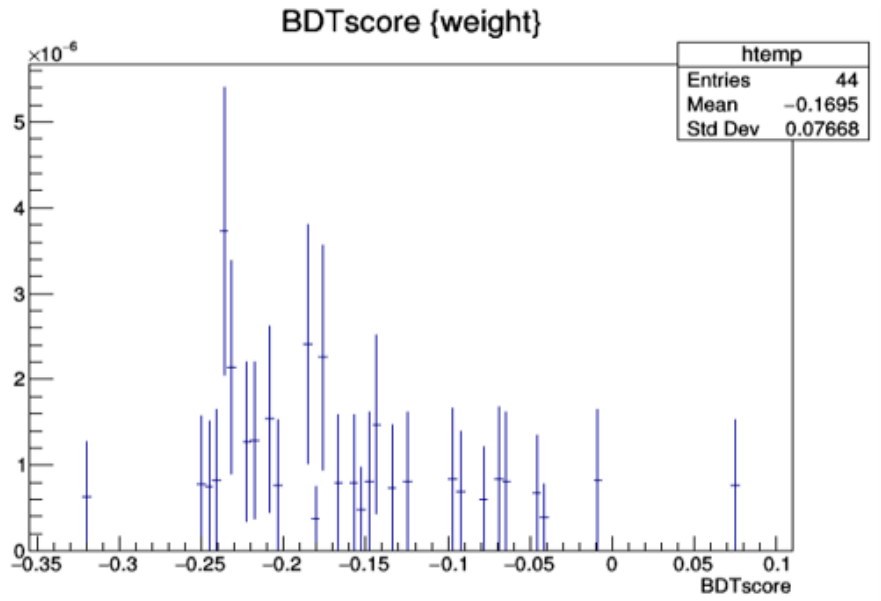


Figure 29: Z eweak



BDT score comparison - signal

Figure 30: qqZH

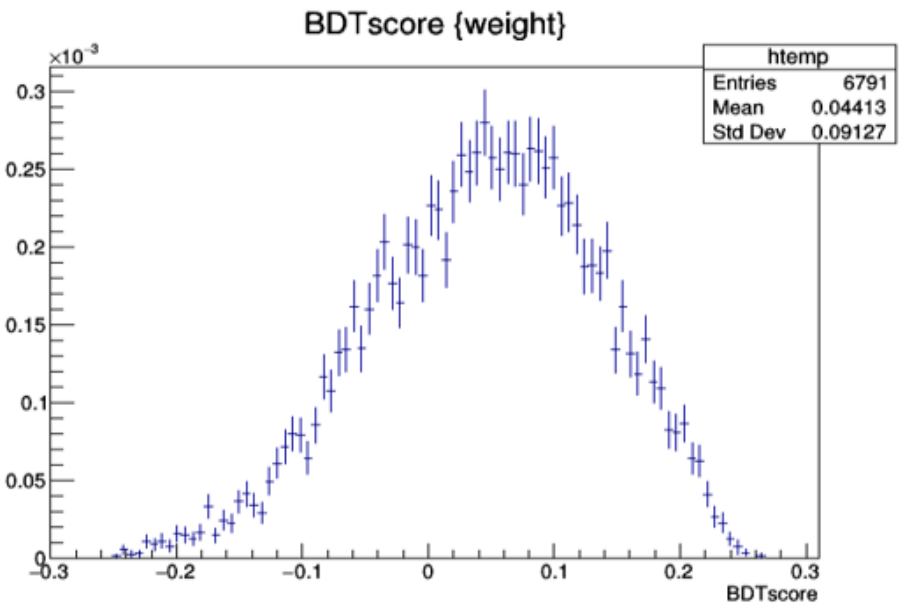
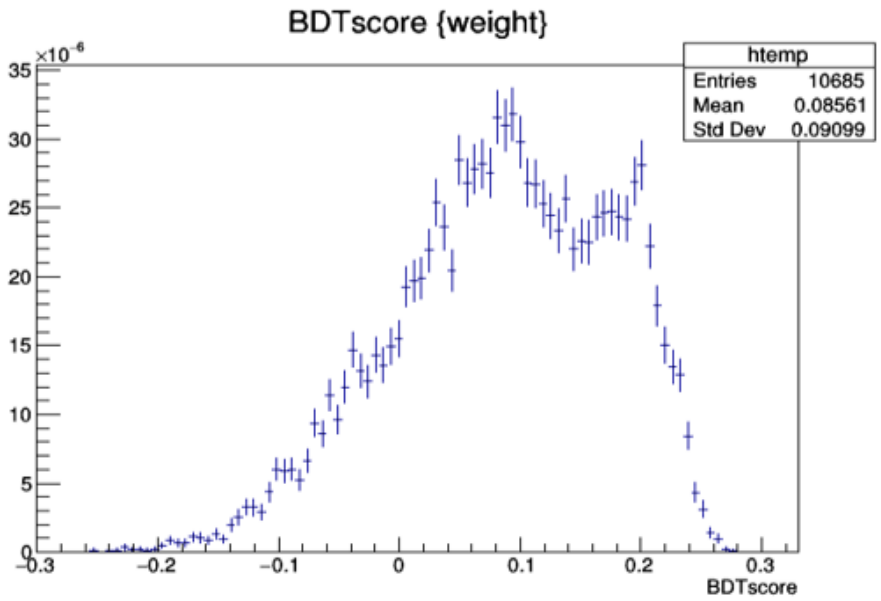


Figure 31: ggZH



AOB

- transformations=I,D,P,G,D
- nuovo training con depht alta (si osserva overtraining o no?)
- ———————
- dubbio sulla media, a denominatore non dovrebbe esserci la somma dei pesi?
- mc16a, mc16e ,...
- gli ultimi due plot di TMVA, BDT e l'ultimo, non hanno funzionato
- correzioni professore
- come vedere tutti i fondi e il segnale sullo stesso istogramma?