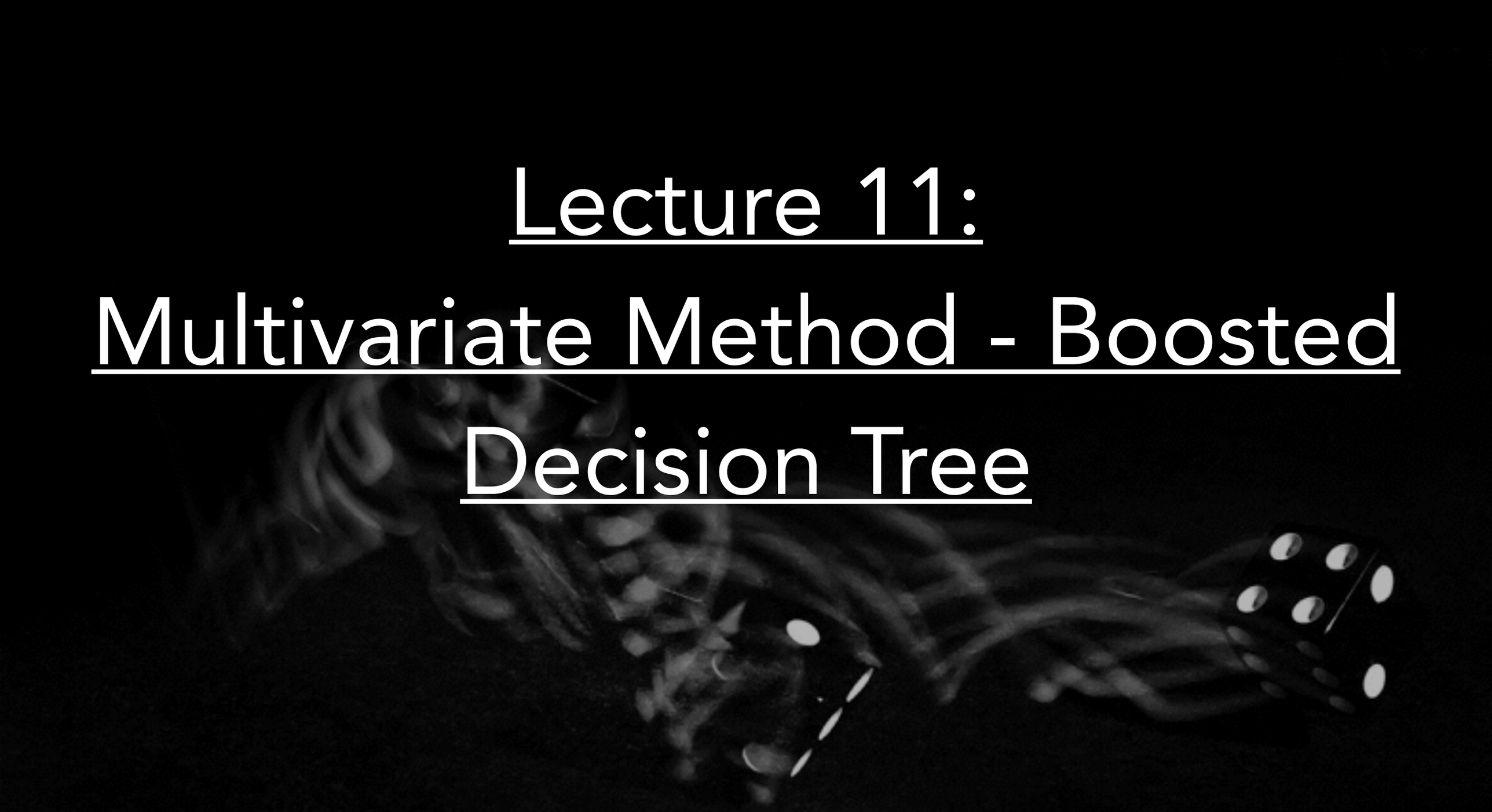
Lecture 11:
Multivariate Method - Boosted Decision Tree



D. Jason Koskinen
koskinen@nbi.ku.dk

Advanced Methods in Applied Statistics
Feb - Apr 2016

Lecture Material Credits:
H. Voss (MPIK) , TMVA group

Photo by Howard Jackman

"Simple" Problems

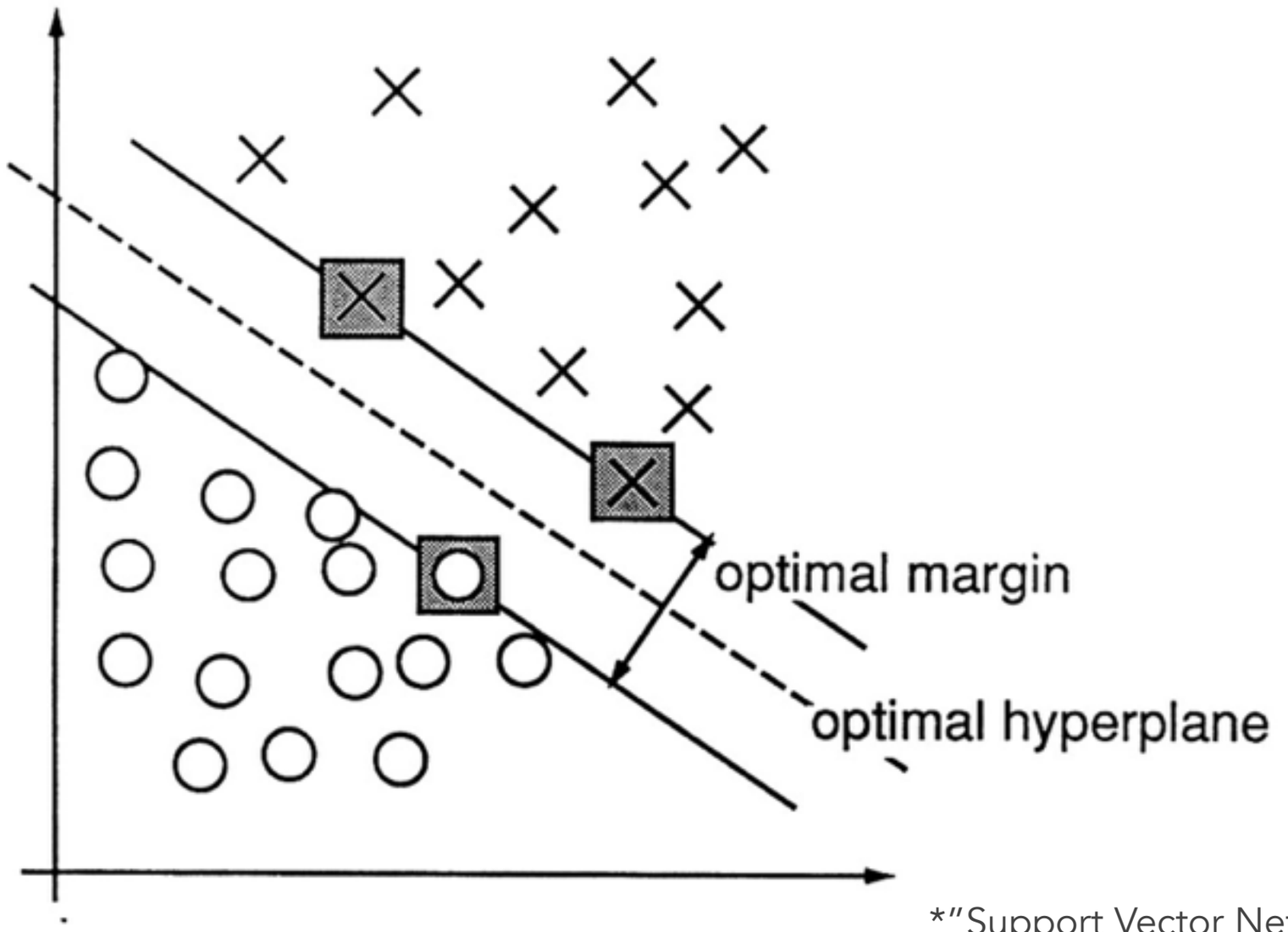
- Using likelihoods and Neyman-Pearson lemma to separate background from signal is not always feasible
  - Likelihood is too complicated for analytic or Monte Carlo evaluation
  - High dimensionality makes Monte Carlo computationally expensive
- Data sets which are linearly separable in variables, e.g. between signal and background, have useful tools for doing such a separation (Fisher Discriminant).
- For linear and non-linear classification scenarios and/or where the available separators are weak, there is a class of multivariate tools
  - k-Nearest Neighbor
  - Random Forest
  - Artificial Neural Networks
  - Support Vector Machine (can be a linear regression classifier too)
  - **(Boosted) Decision Trees**
  - etc.

Supervised Learning Algorithms

- We might not be able to easily calculate likelihoods or probability distribution functions, but we can separately identify data or generate Monte Carlo which is known signal and known background
- Use the known signal/background as training samples for a learning algorithm to classify events as signal/background based on only the available variables
- A user provides the events, true classification data sets, and variables, and the machine algorithm (hopefully) produces an inference which can be used on unclassified data to separate signal/background

Support Vector Machine

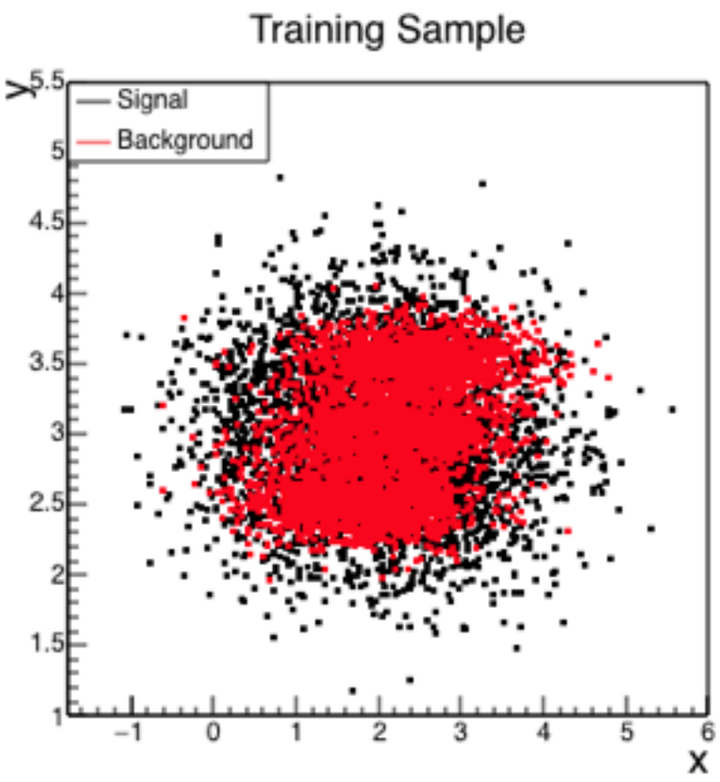
- Depending on the kernel, the SVM maps the data into a higher or alternate dimension space, and makes classifications via hyperplanes

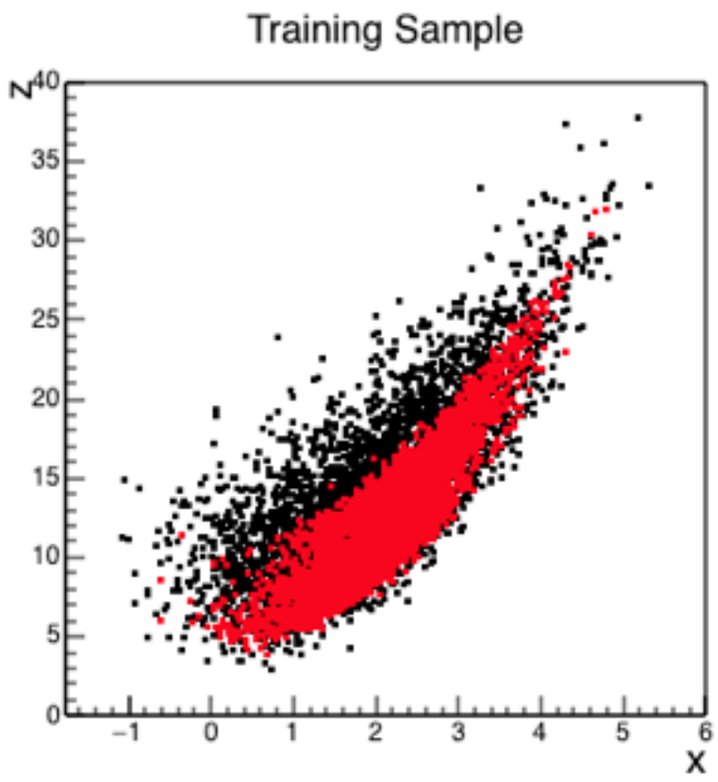


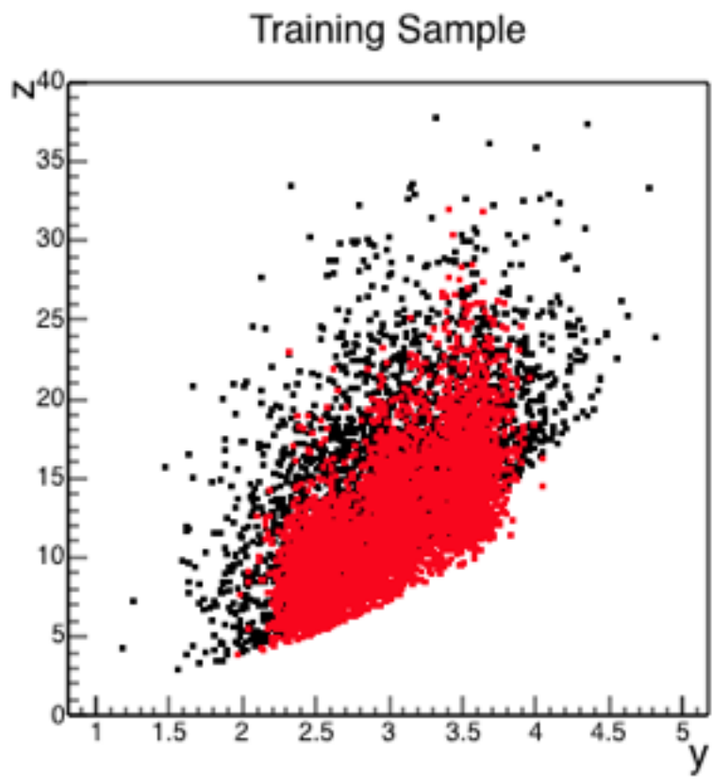
*"Support Vector Networks", Cortes & Vapnik

Training Samples

- Below are events in only x, y, and z for some class of signal and background, which are not clearly too separable via straight cuts
  - Machine algorithms can ‘see’ in higher dimensions very quickly
  - This semi-simple example with a boosted decision tree can get 88% signal efficiency at 1% background contamination

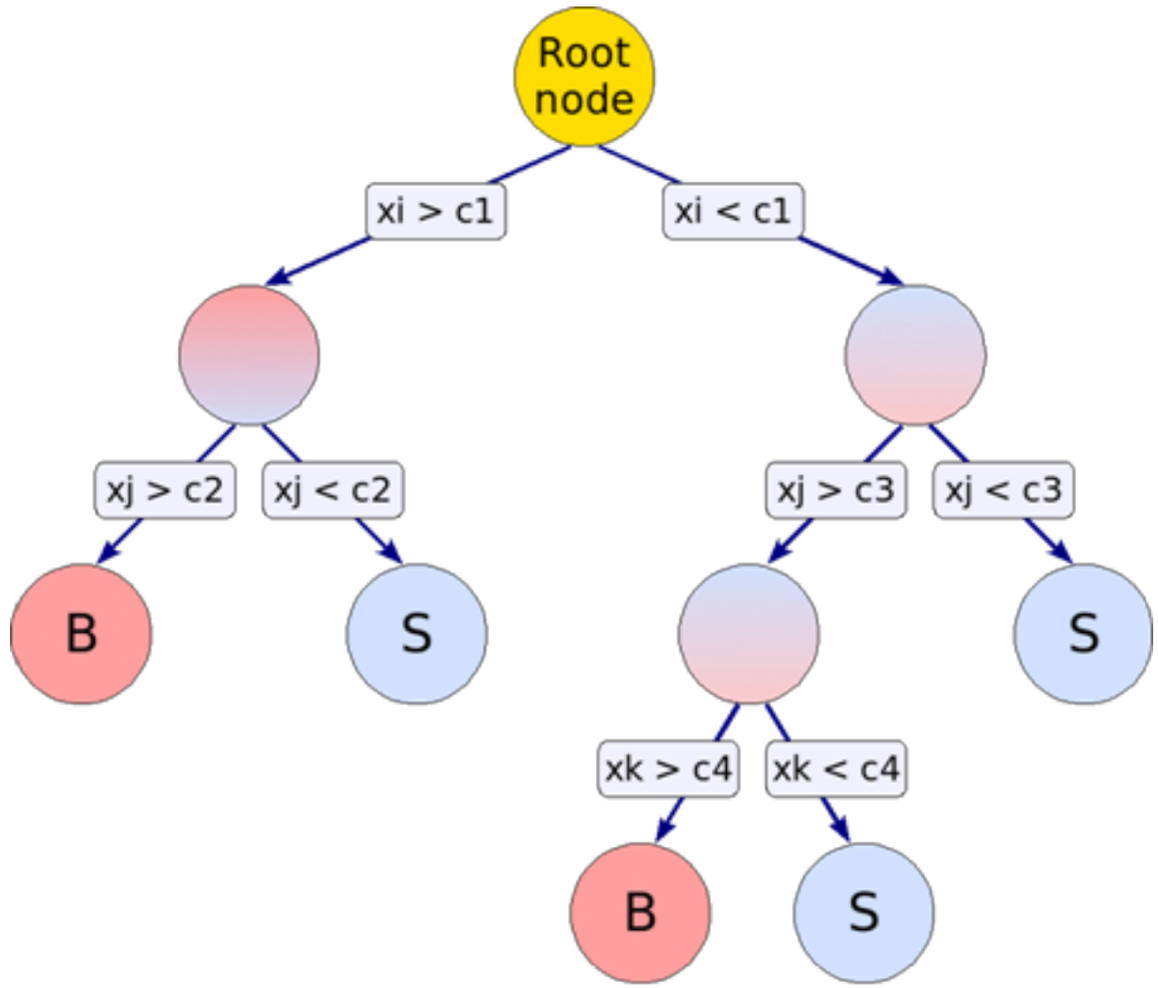






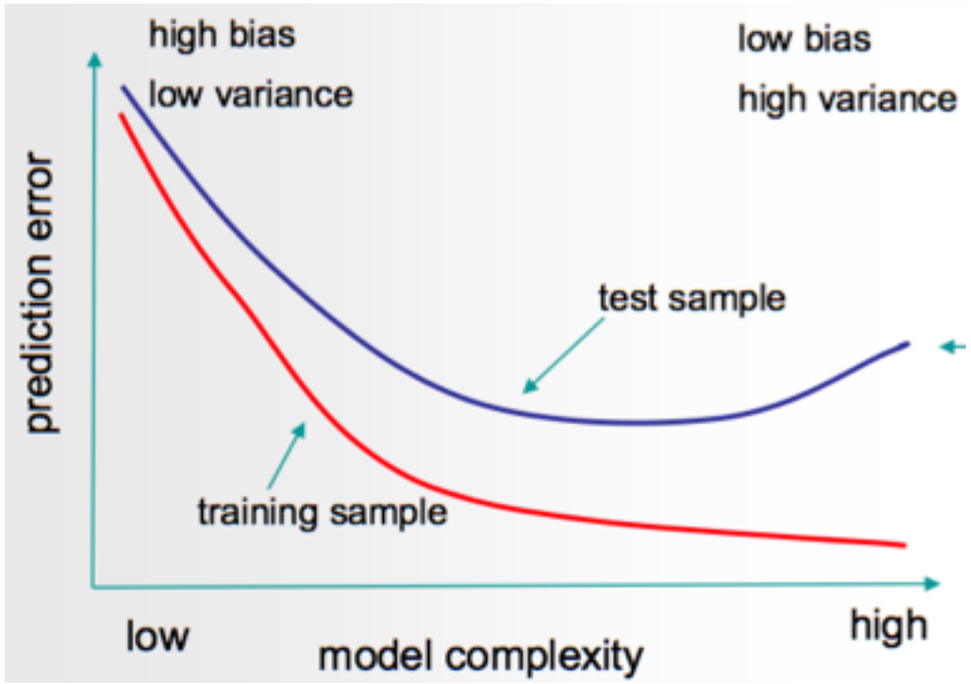
Decision Tree

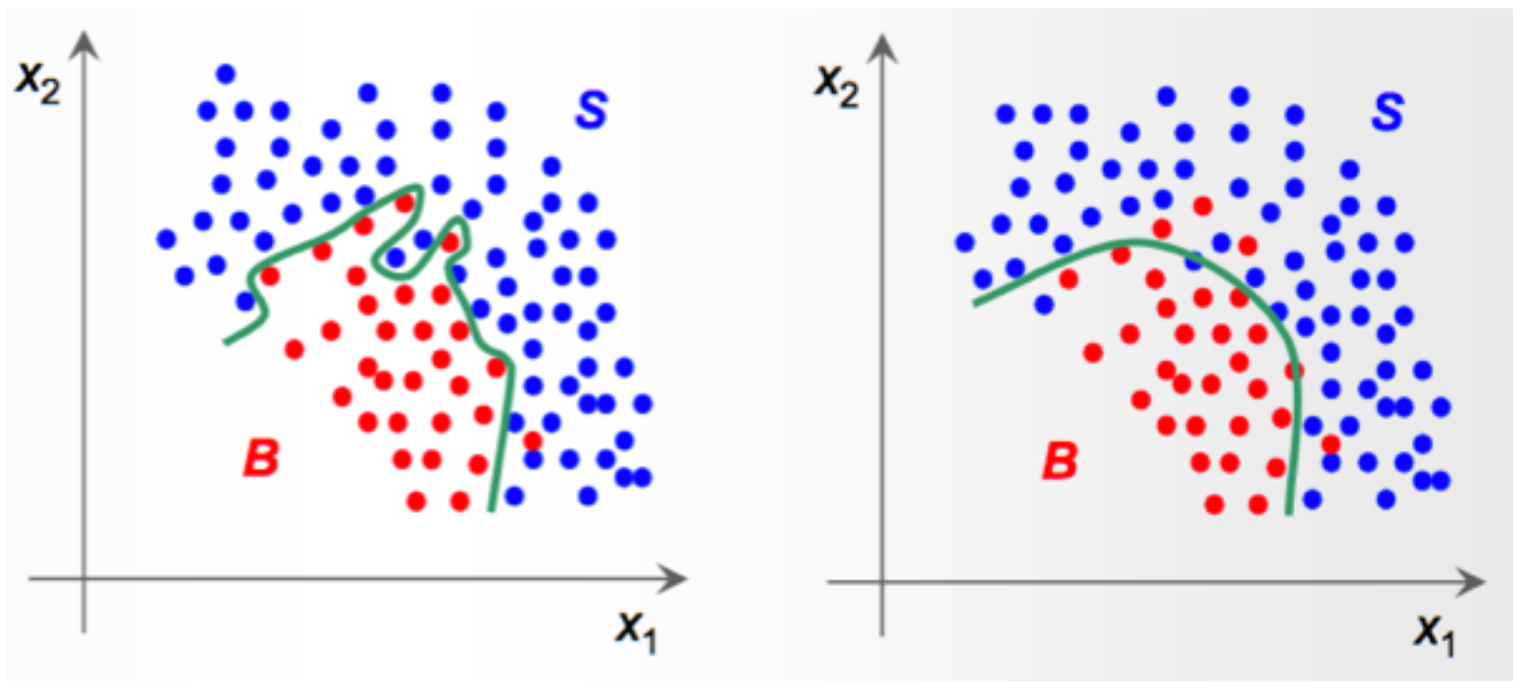
- Decision Tree: Sequential application of cuts splits the data into nodes, where the final nodes (leaves) classify an event as signal or background
  - Easy to visualize and interpret
  - Resistant to outliers in data
- Disadvantage is that statistical fluctuations in training create instability



Overtraining

- Machine Learning algorithms can be overly optimized wherein statistical fluctuations from the training data are wrongly characterized as true features of the distributions
  - Deficit of training data statistics versus number of variables or complexity
  - Model flexibility, e.g. many free parameters

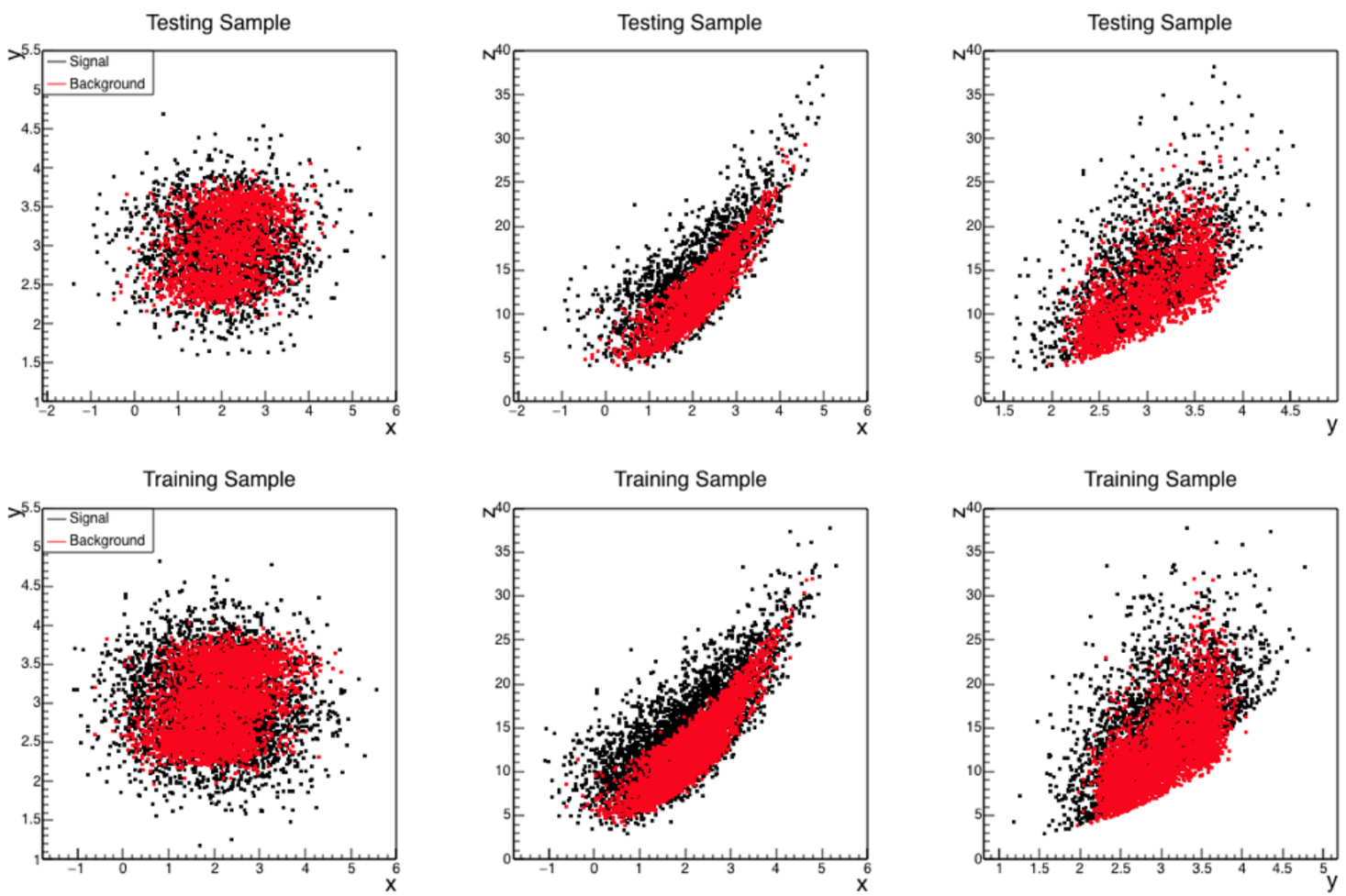




*H. Voss (MPIK)

Testing & Training

- A common way to check over-training for any machine learning algorithm is to test the learned inference classification on a statistically independent data set of known signal/background
- Classification should be as similar between the training sample results and testing sample results as statistical fluctuations permit. Significant differences, e.g. in the ROC curves, are a common indicator of over training.

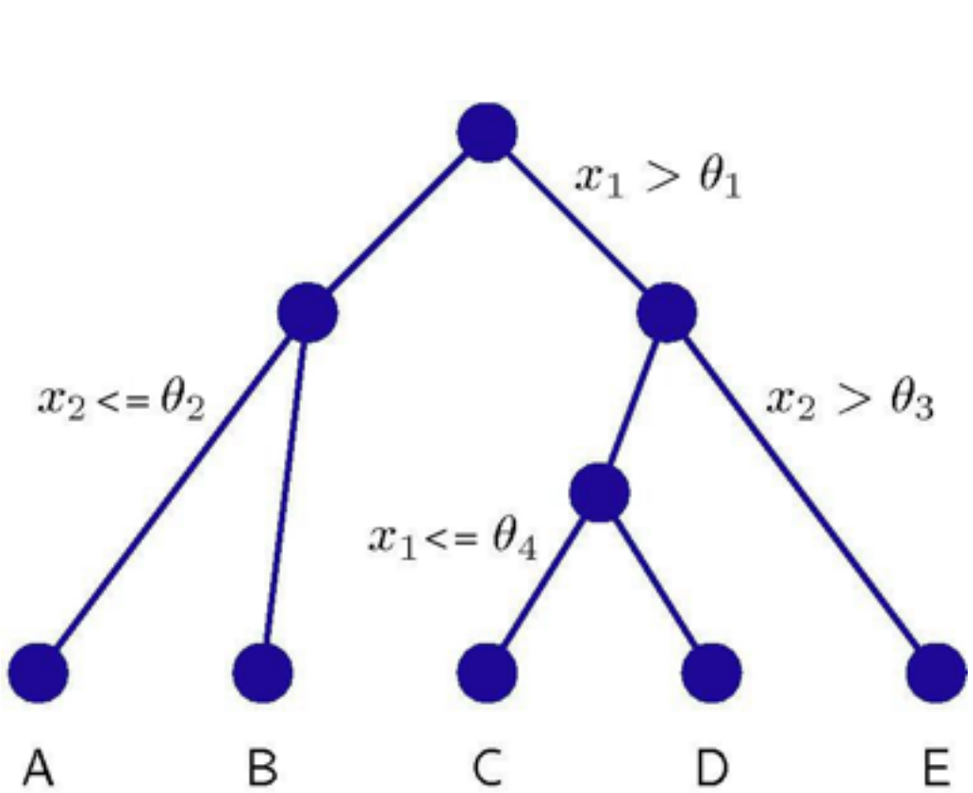


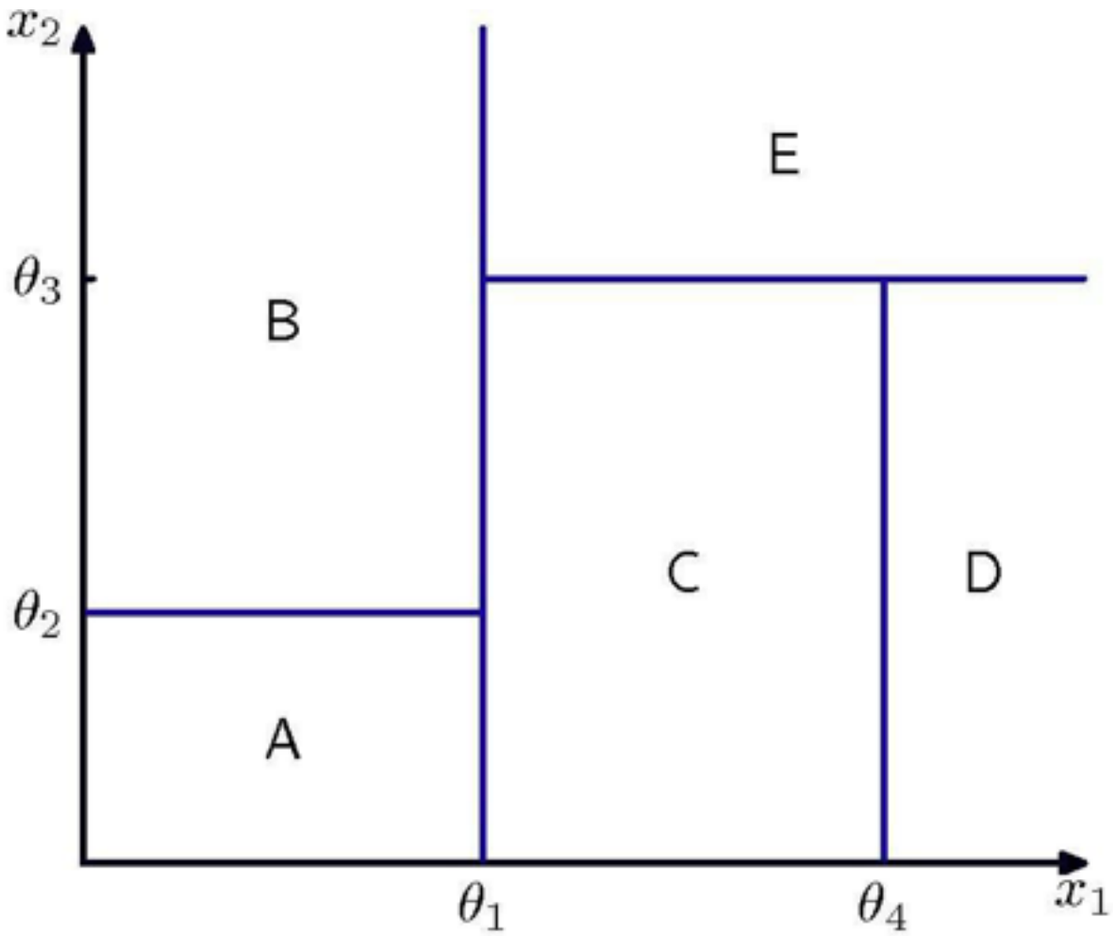
Creating the Tree

- Start with a training sample and split using a variable that gives the best separation (for some definition of ‘best’, e.g. Gini-index)
- Continue splitting until some threshold is met:
  - Statistics per node
  - Number of nodes
  - Depth of decision tree
  - Further splits fall below separation threshold
- Events that fall in the final nodes are classified as signal/background via some metric (binary classification, sig/bkg probability, etc.)

Decision Tree Walkthrough

- The decision tree (left) and the 2D space (right) which represent the different areas of x1 and x2 classified as signal/background regions by the decision tree



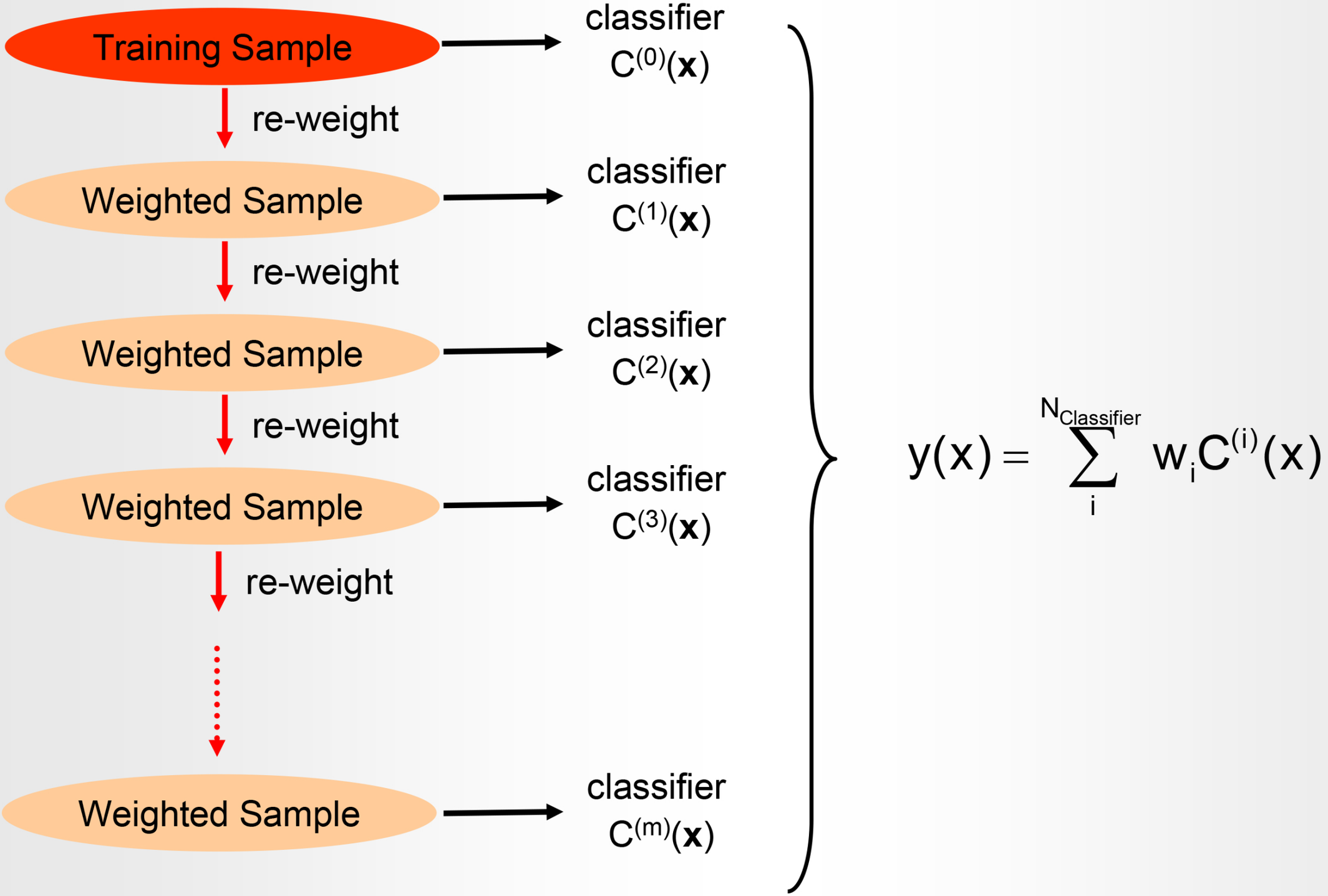


*J. Therhaag

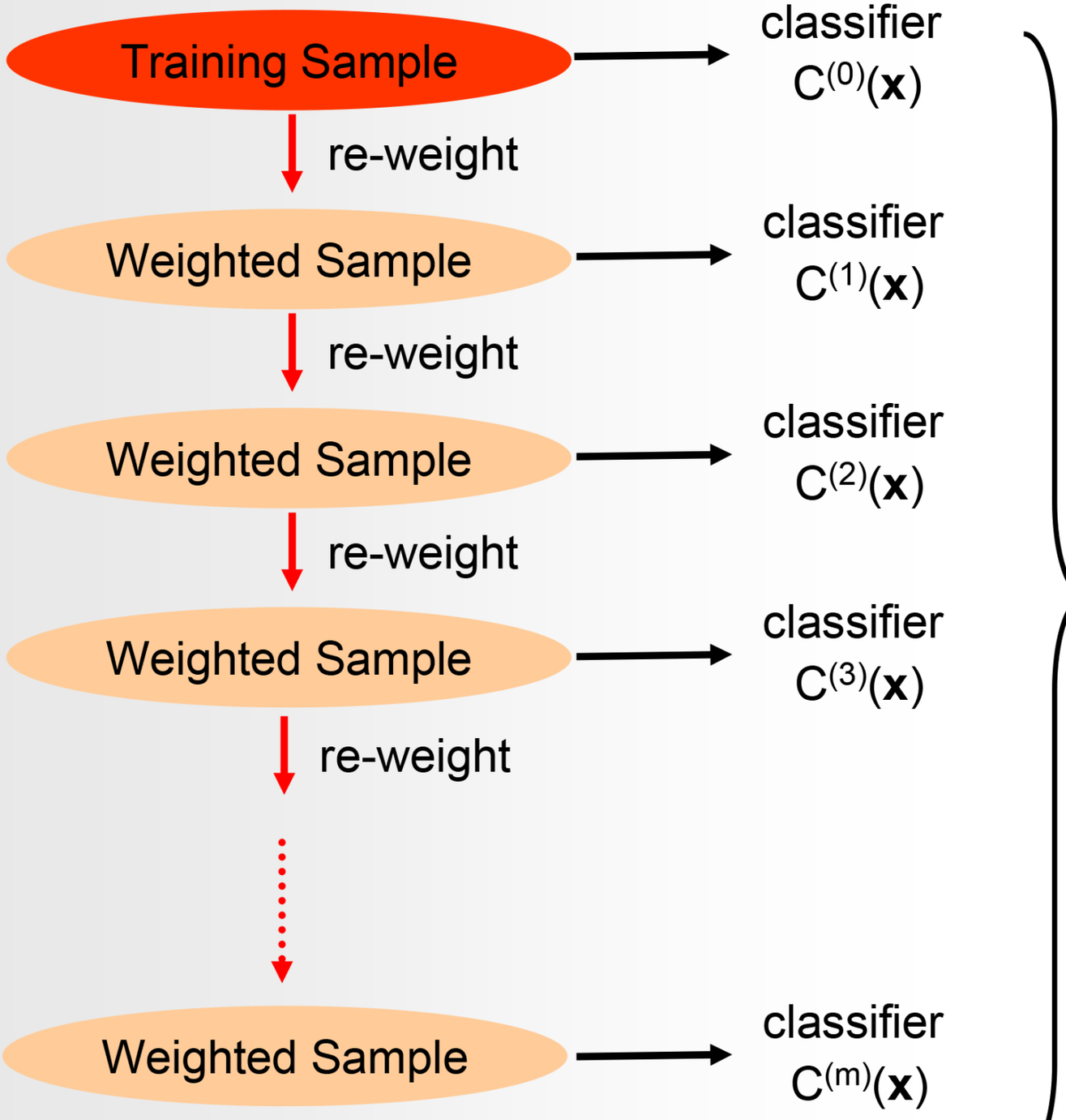
Decision Tree Overtraining

- Similar to machine learning algorithms, a decision tree can be overtrained. Specifically, it can be very sensitive to statistical fluctuations in the training sample.
  - High statistics training samples can minimize the impact, but never remove it
  - Removing nodes with low separation power, thereby reducing the complexity
- Could generate multiple trees and combine them to increase separation power and decrease overtraining, but the decision process would create identical trees for the same training sample
- Common solution is to use Boosting

Boosting


Training Sample
classifier
$C^{(0)}(\mathbf{x})$
re-weight
Weighted Sample
classifier
$C^{(1)}(\mathbf{x})$
re-weight
Weighted Sample
classifier
$C^{(2)}(\mathbf{x})$
re-weight
Weighted Sample
classifier
$C^{(3)}(\mathbf{x})$
re-weight
Weighted Sample
classifier
$C^{(m)}(\mathbf{x})$
$y(x) = \sum_{i}^{N_{Classifier}} w_i C^{(i)}(x)$

Adaptive Boosting (AdaBoost)



- AdaBoost re-weights events misclassified by previous classifier by:

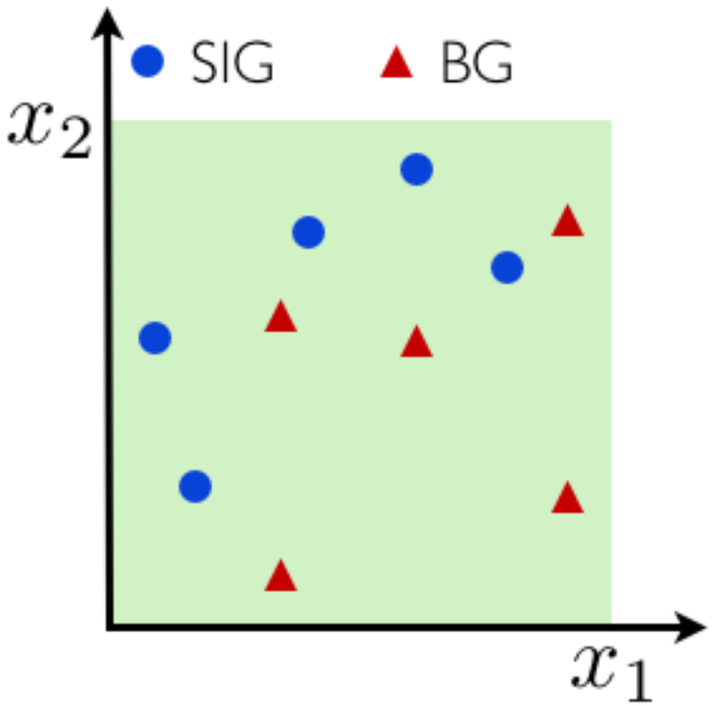
$$\frac{1-f_{err}}{f_{err}} \quad \text{with :}$$

$$f_{err} = \frac{\text{misclassified events}}{\text{all events}}$$

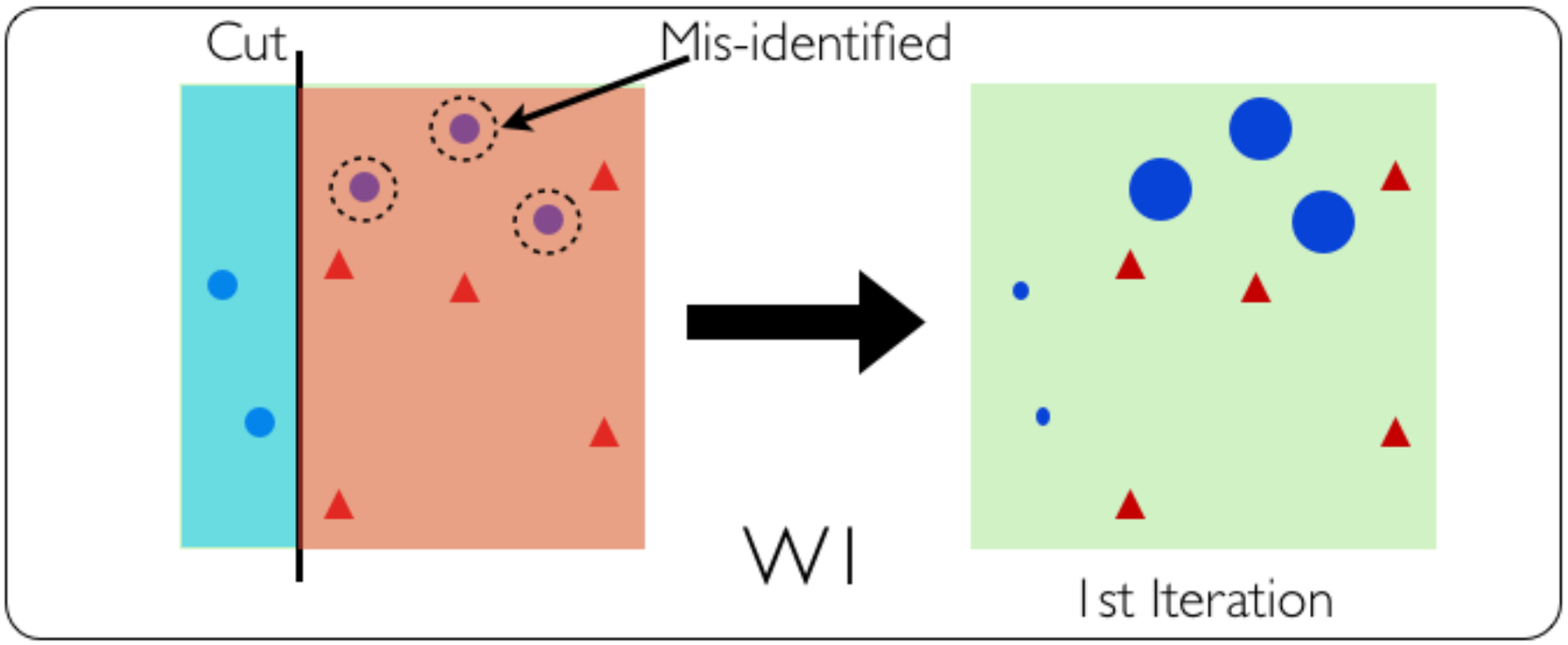
- AdaBoost weights the classifiers also using the error rate of the individual classifier according to:

$$y(x) = \sum_{i}^{N_{Classifier}} \log\left(\frac{1-f_{err}^{(i)}}{f_{err}^{(i)}}\right) C^{(i)}(x)$$

Adaptive Boosting Visually



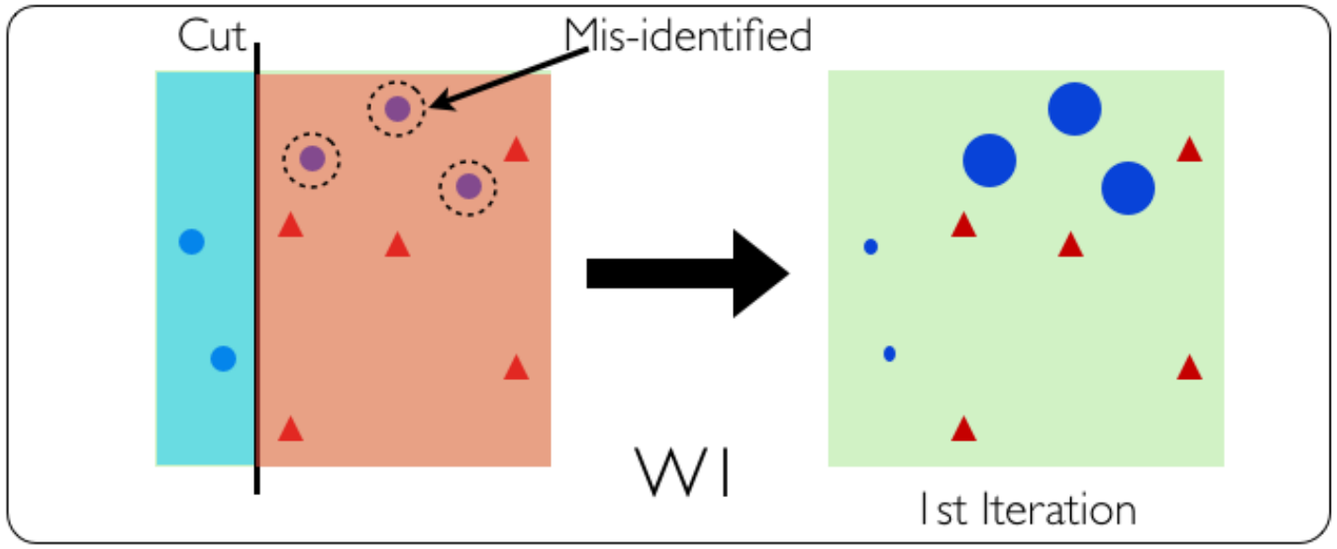
<Simple 2-D toy example>
Hypothesis : Straight Cuts
Multiple Iterations
Boosting : Give different weights
when (mis)classified

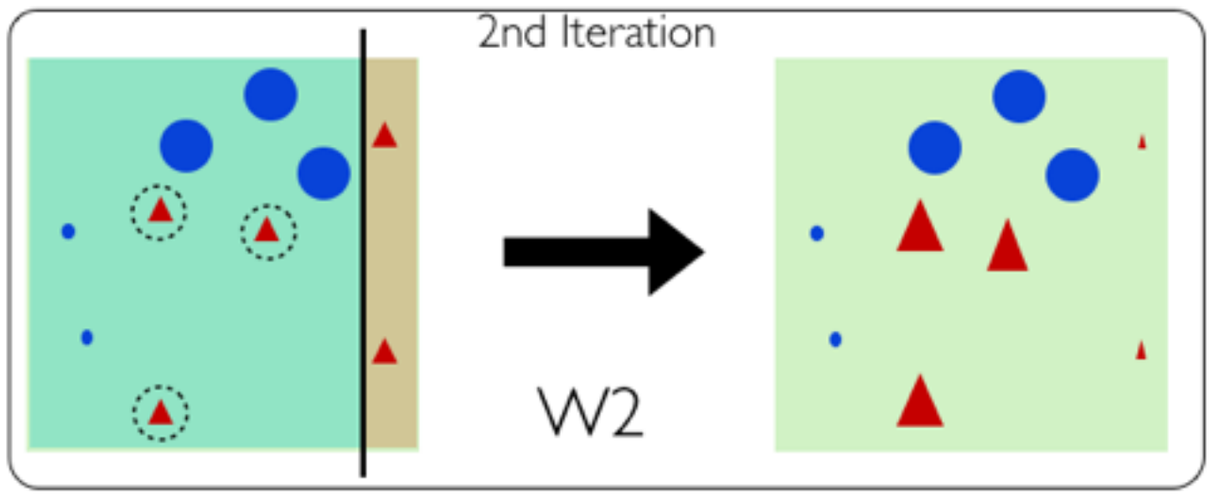


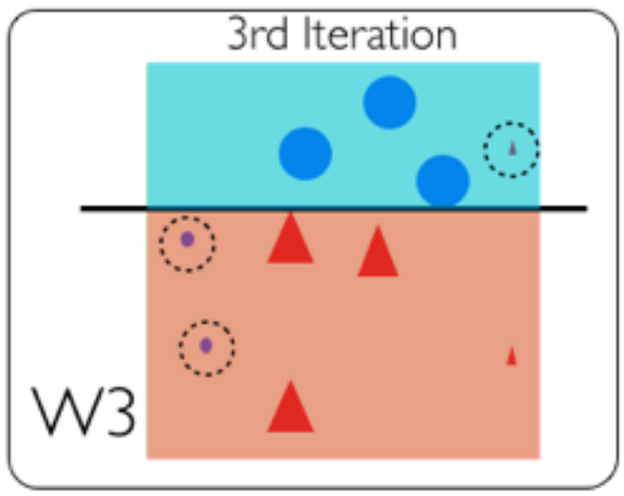
*AdaBoost by Y. Freund & R. E.Schapire

Boosted Decision Trees

- Past the first one, each iterative boosted decision tree (classifier) is trained on the ‘same’ events. But now, the events have weights according to whether they were previously wrongly classified.

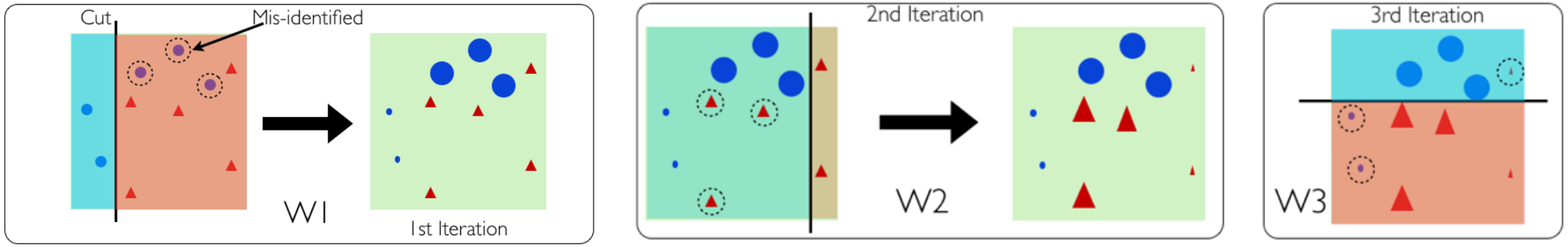


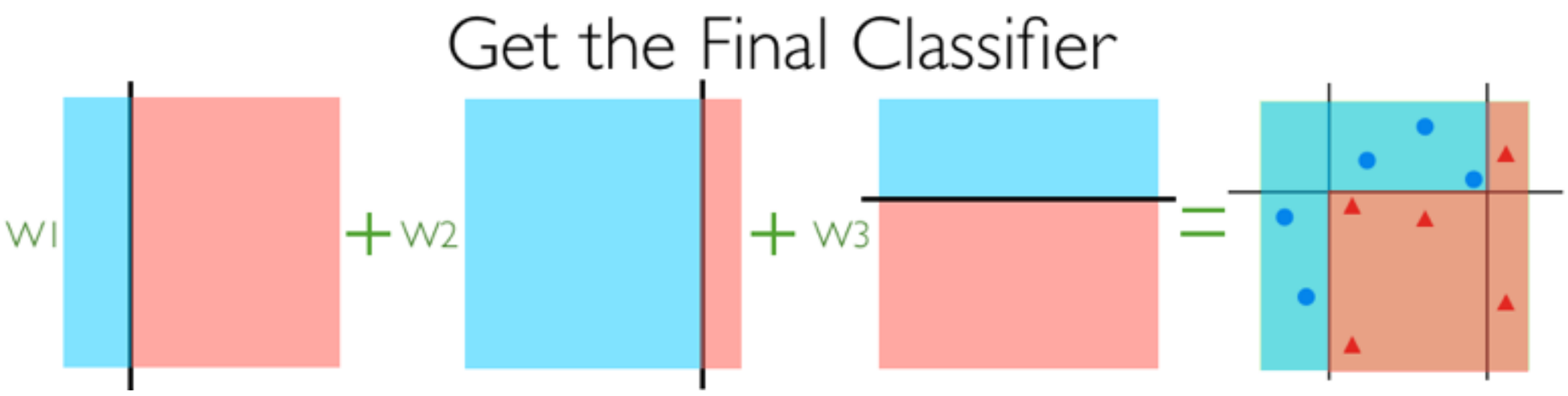




Boosted Decision Trees

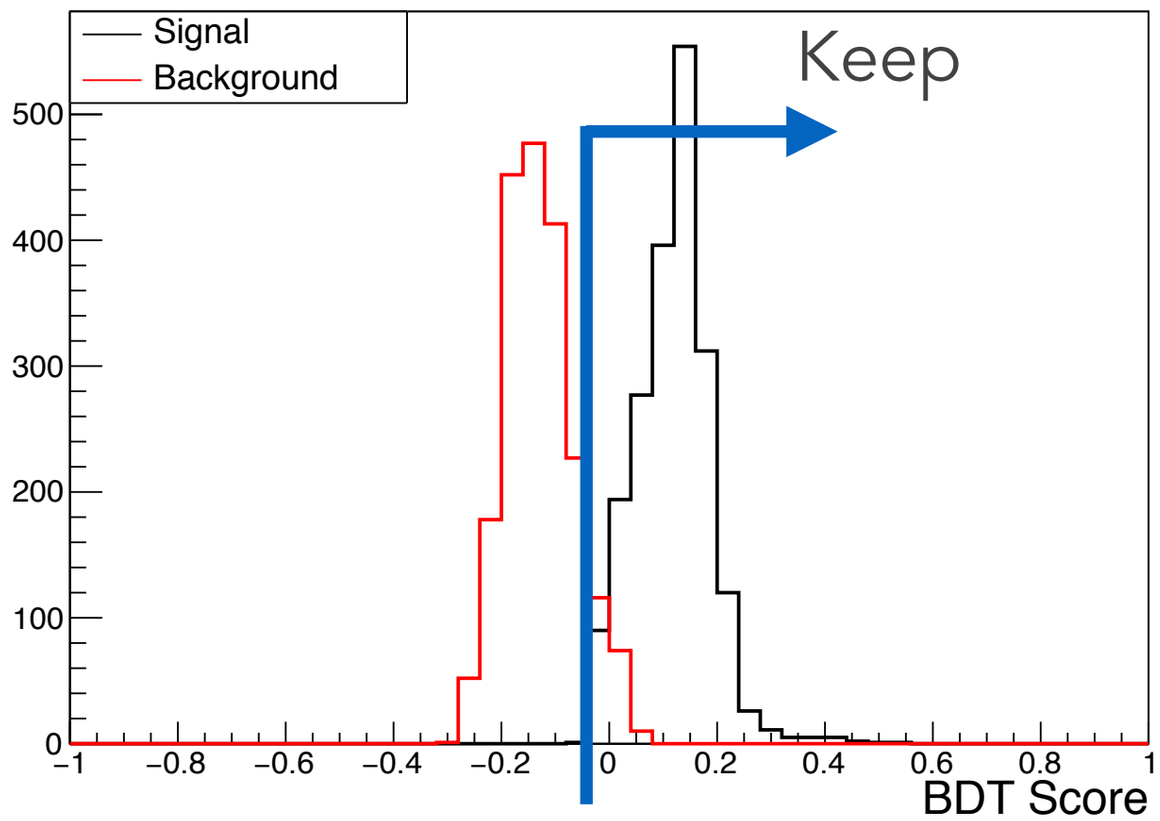
- The combined classifier is the weighted average from all trees for the different regions
- Works very well “out-of-the-box”





Boosted Decision Tree Classifier

- After training, and hopefully testing, the BDT can generate a score when run over new data that allows signal/ background separation
  - More negative values are background
  - Place a cut at some score to get desired purity and efficiency



BDT Comments

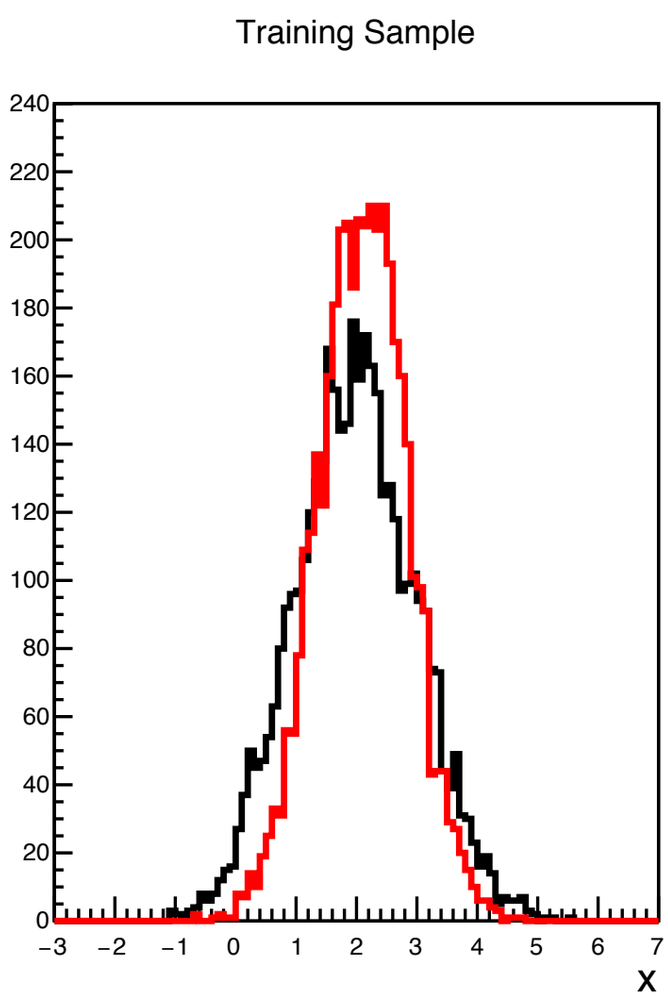
- It is common to throw an absurd number of variables into a BDT and have it signify the variables of importance, i.e. the 'kitchen sink' approach. The more variables used in any supervised learning algorithm, the more difficult it is to debug when something goes wrong, e.g. user error.
- The number of nodes, variables, events, and depth of each tree can influence the classification outcome. Because BDTs are generally fast to train, play around with the settings/options to see the effects.
- Ensure that the variables used in training match the distribution shapes in data. Poor variable agreement will bias the BDT, and if the BDT uses many variables it can be hard to notice that a problem exists.

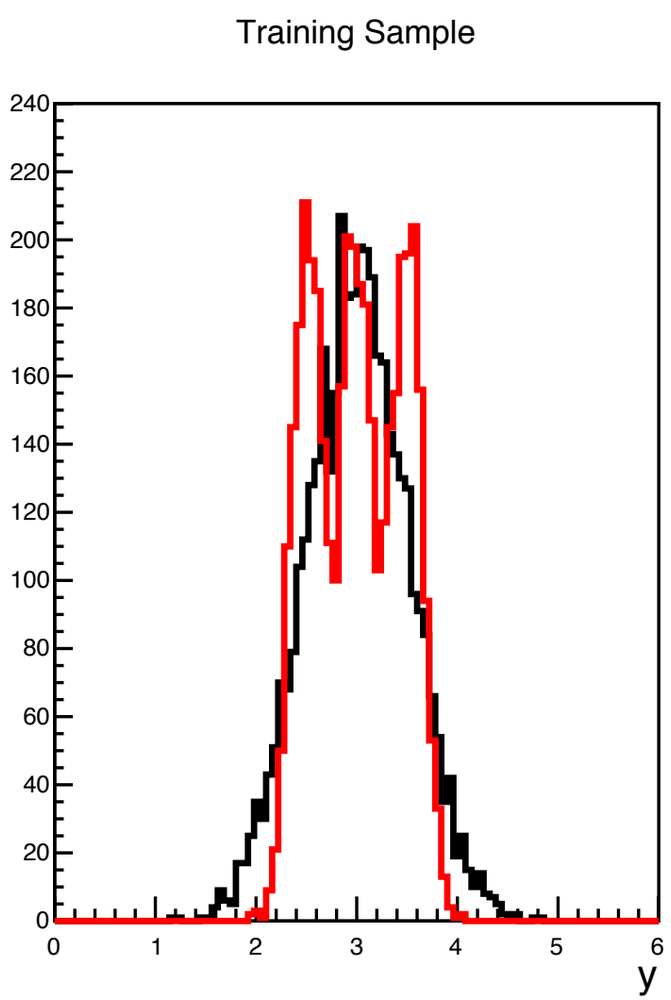
Exercise #1

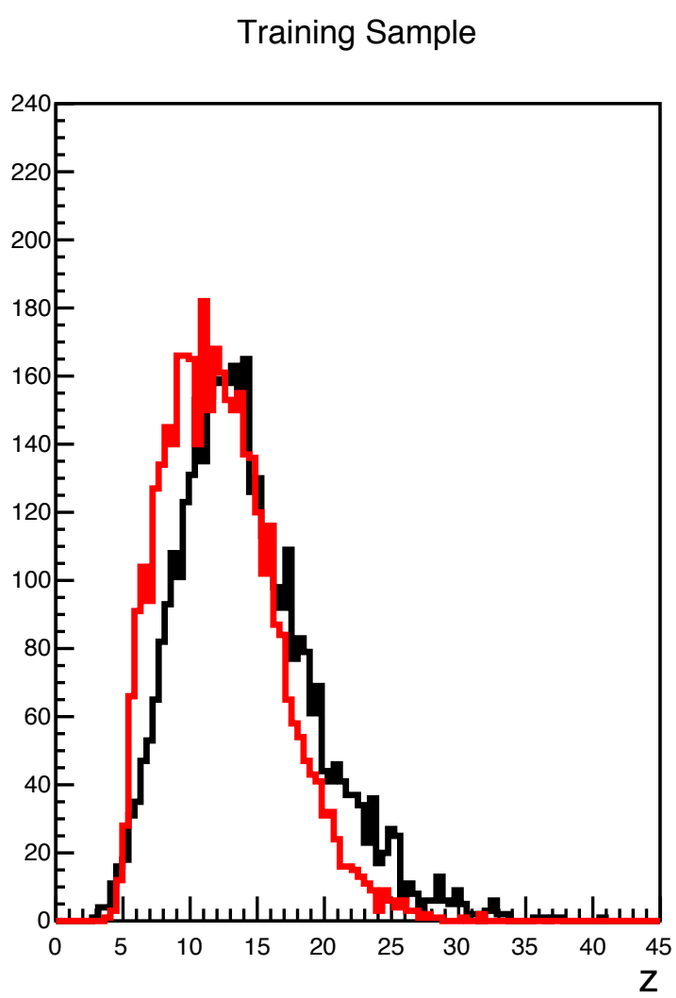
- On the class webpage there are 4 files which include signal and background for training and testing of a learning algorithm (nominally a boosted decision tree)
- The data is generated as an unknown function of three variables
- Plot the three variables for the signal and background samples for training:
  - 1D histograms, one for each variable
  - 2D graphs of the x vs. y, x vs. z, and y vs. z
- After training the BDT, plot the BDT score for the test sample separated by color for the signal and background

Exercise #1 Simple Plots

- Not a whole lot of separation going on here

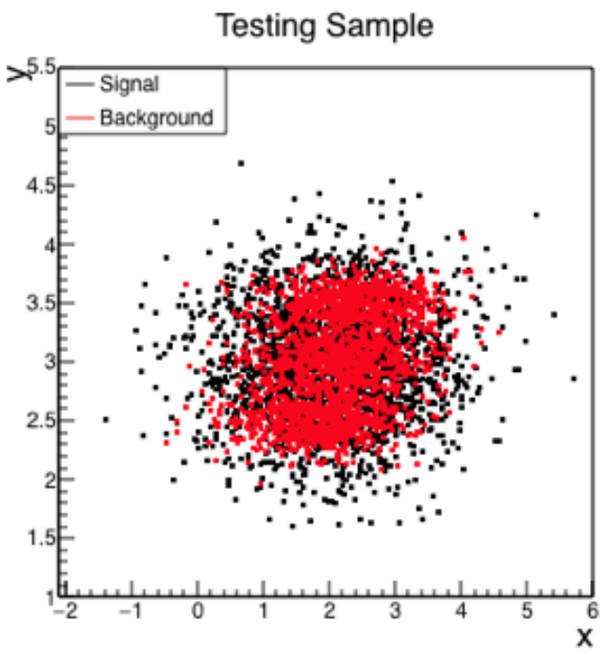


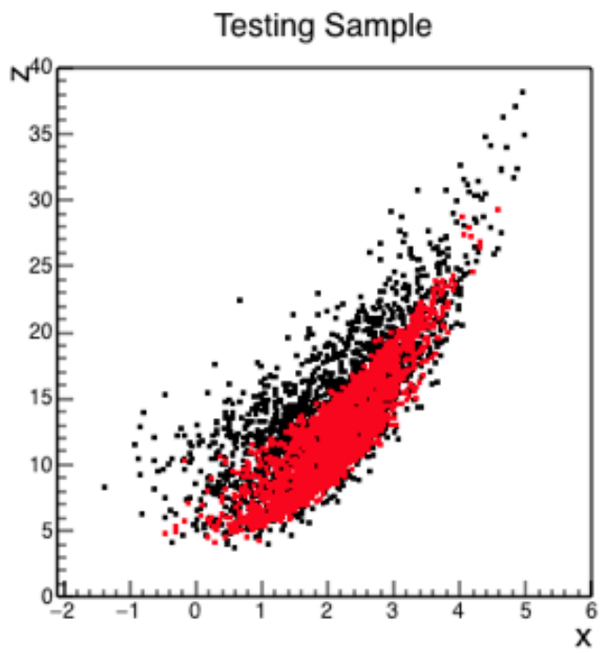


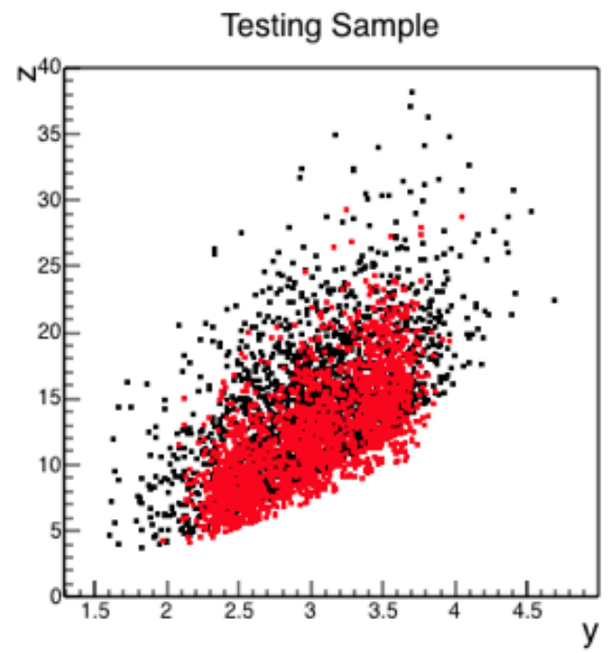


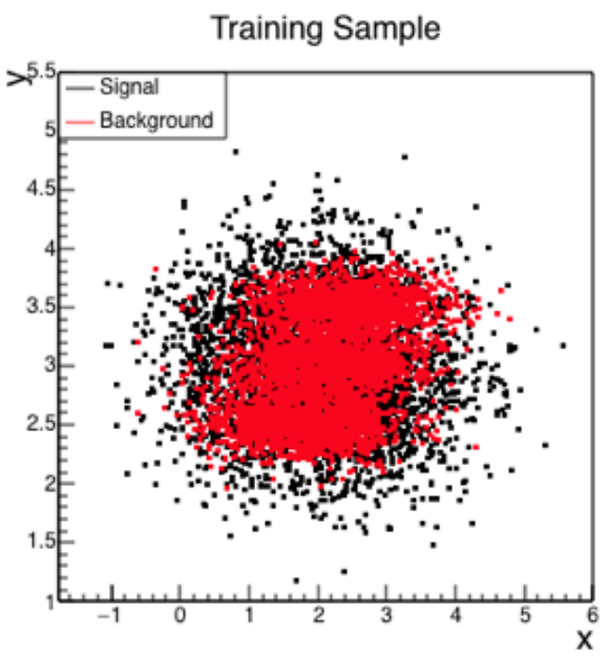
Exercise #1 in 2D

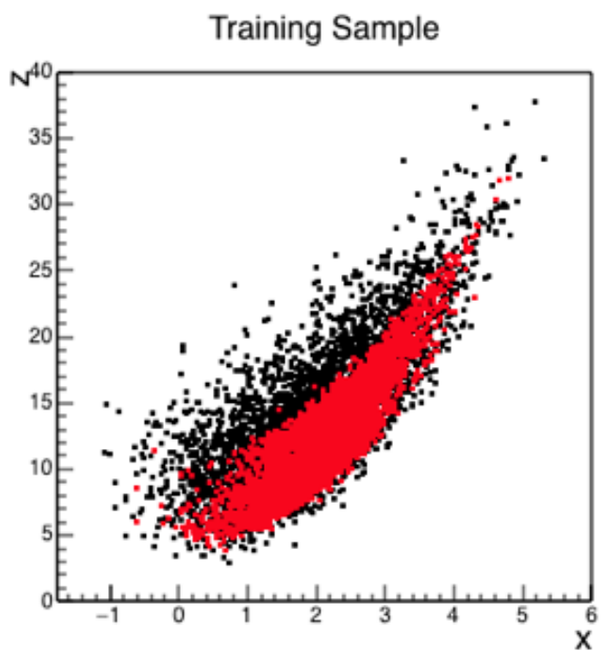
- Not much better in 2D than it was in 1D

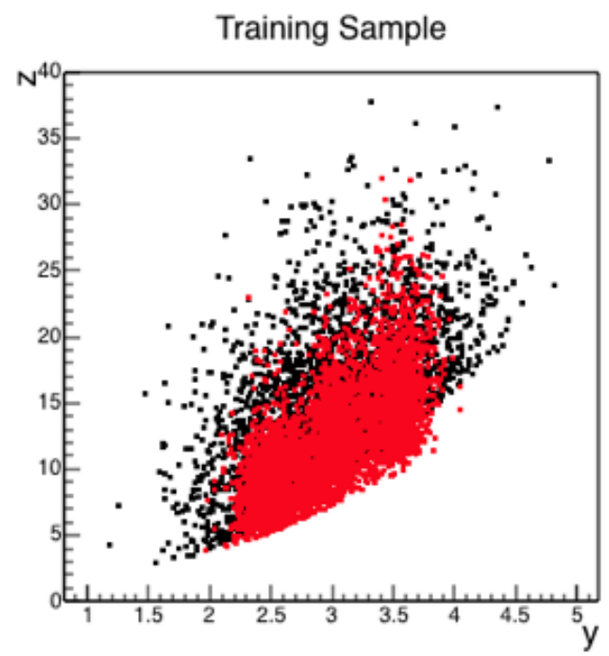




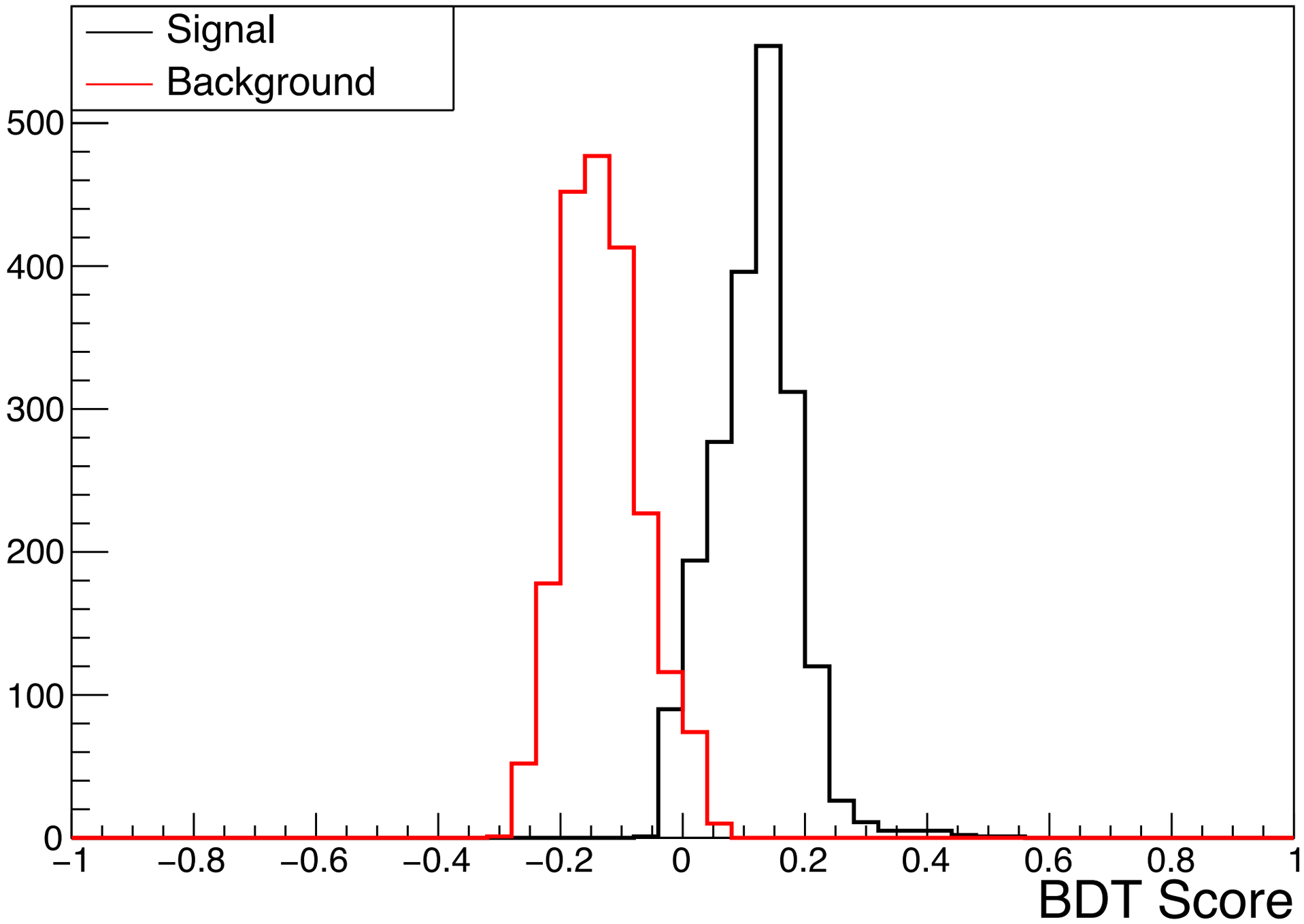








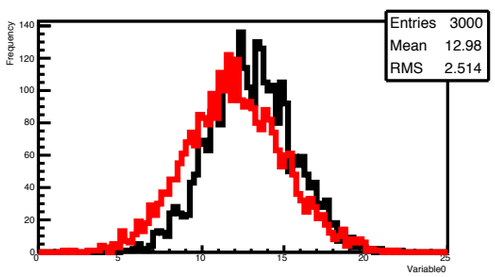
Exercise #1 BDT Score Result

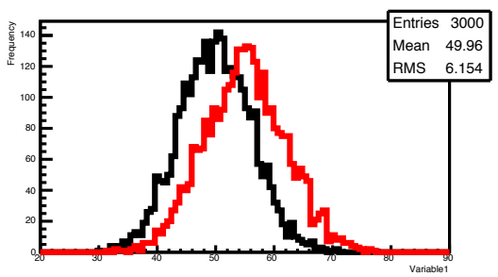


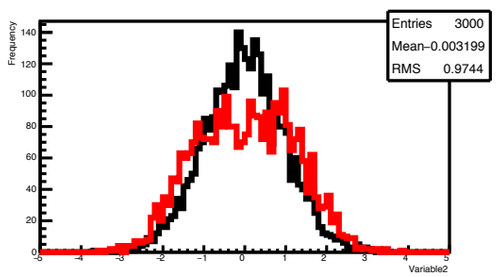
Exercise #2

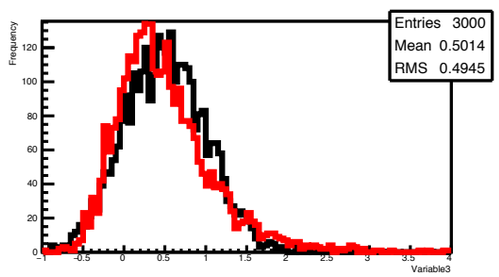
- Now with 16 variables using the a single file on the webpage
- Separate the data from the file into a training and testing sample and repeat
  - Do not use any training data as a test sample
- Train a BDT (or some other machine learning algorithm) and plot the output scores
- Are any variables essentially worthless? What happens when the lowest rank variable is removed from the training?

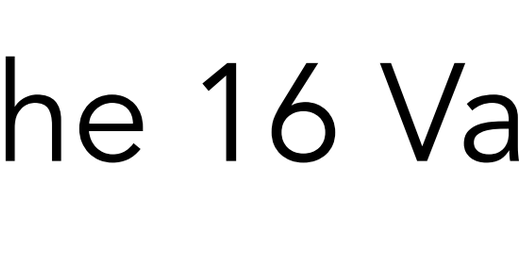
1D Histograms of the 16 Variables

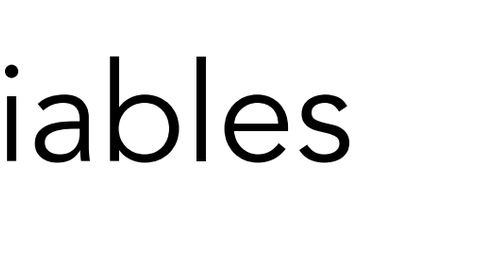


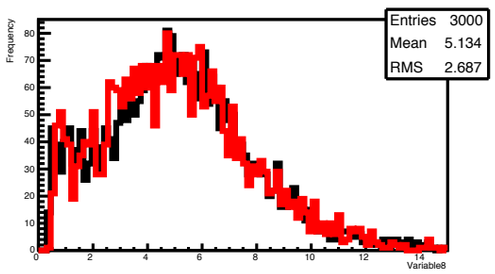


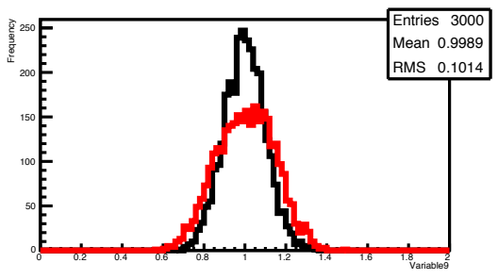


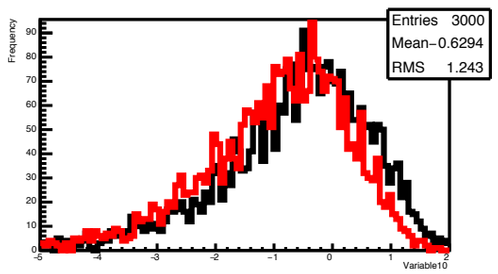


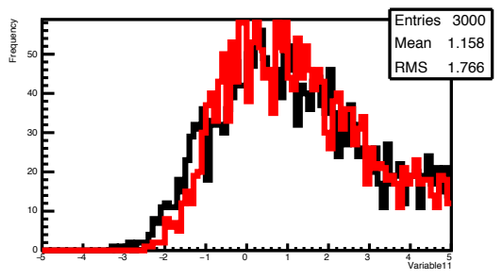


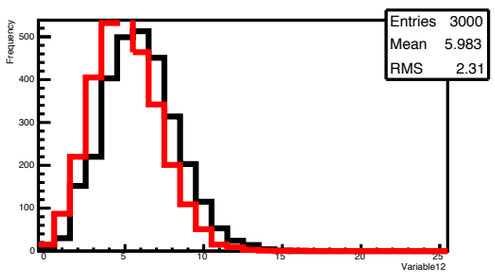


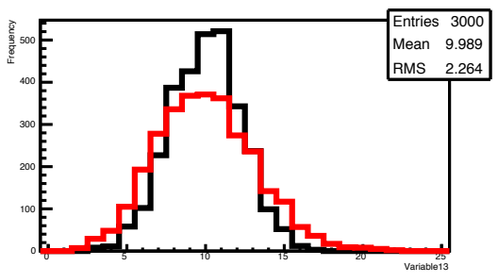


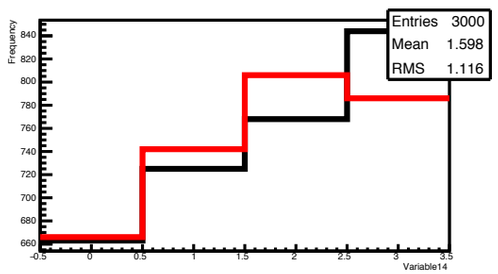


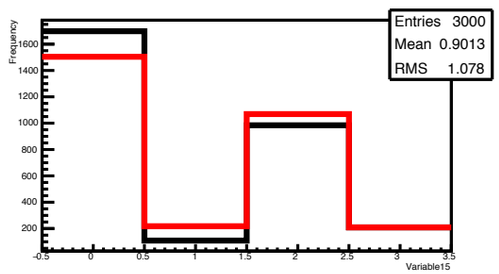




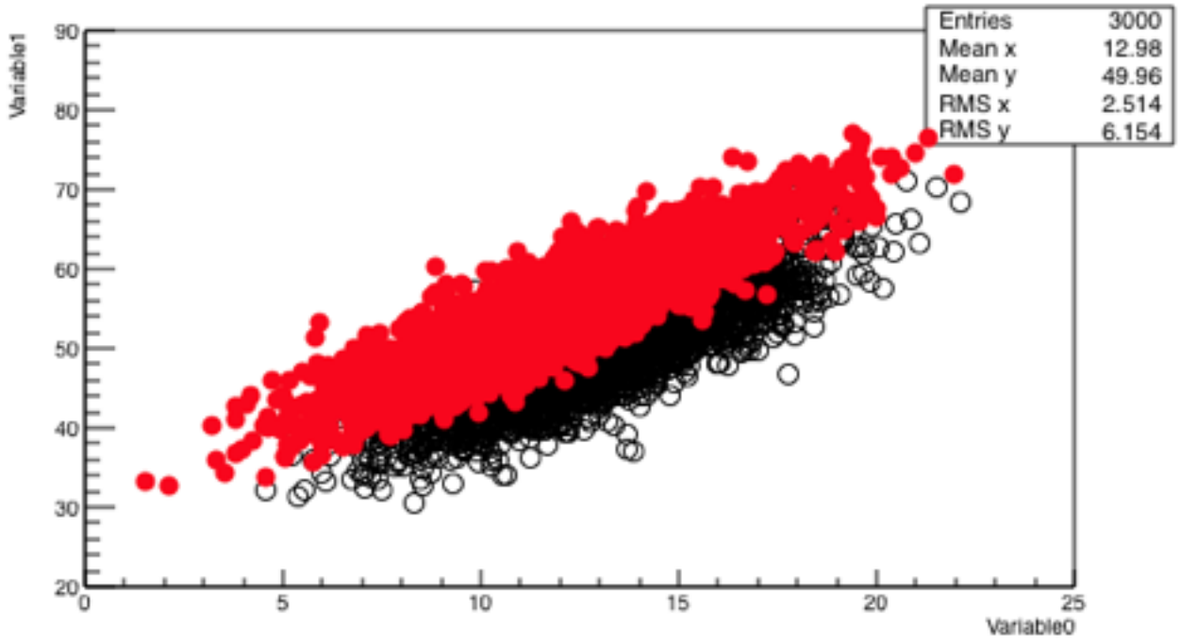


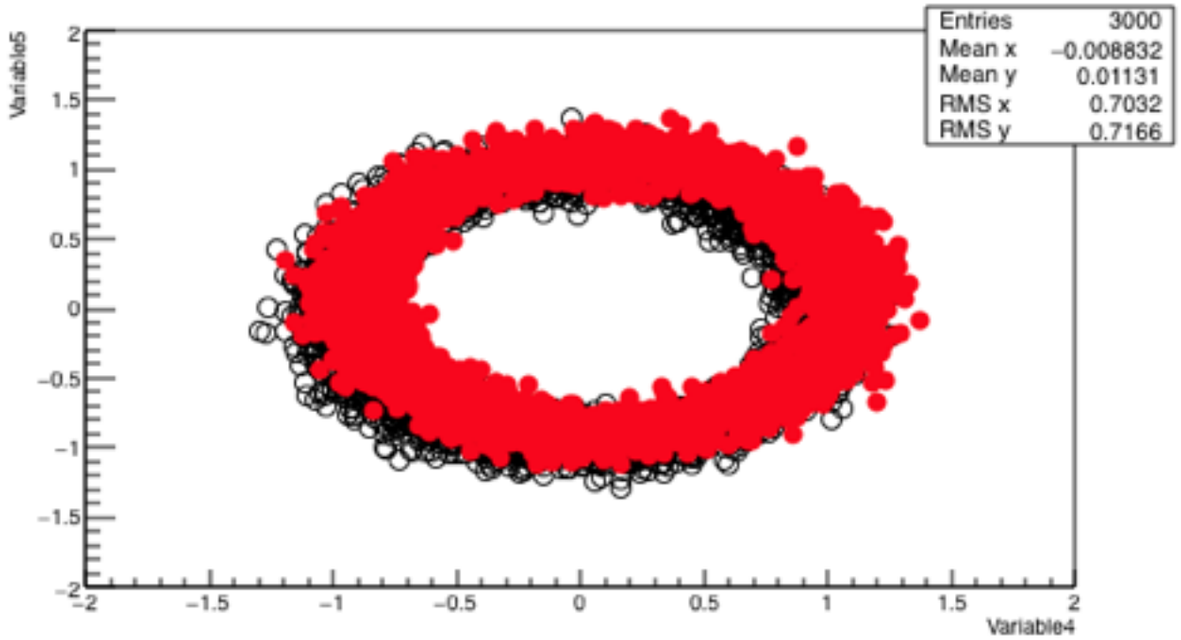


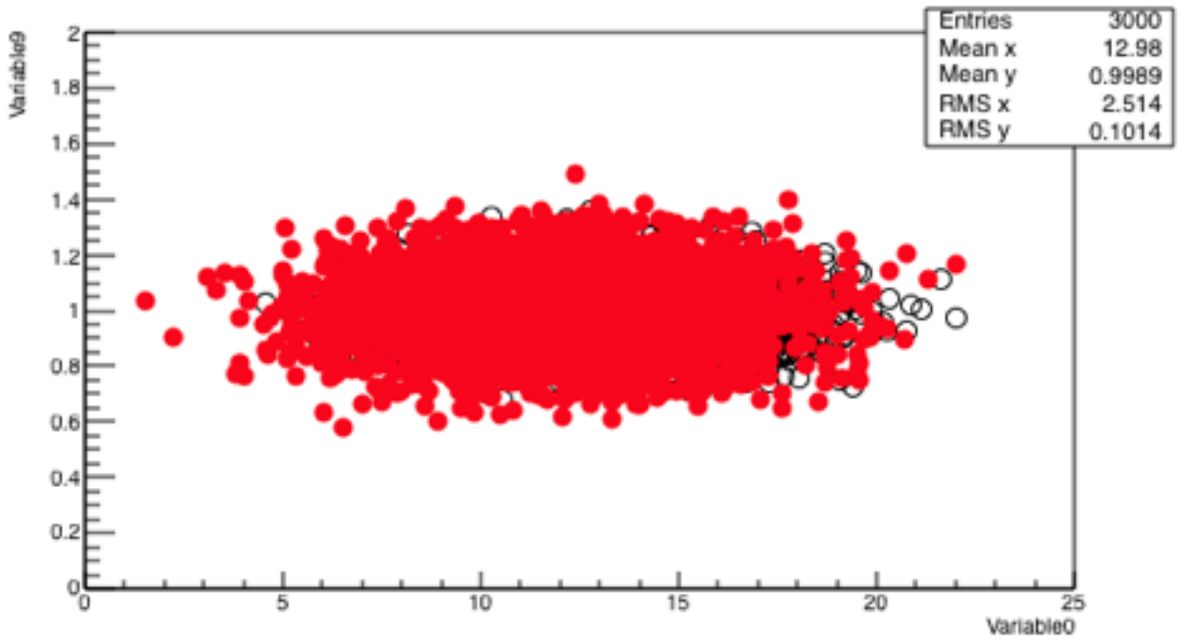


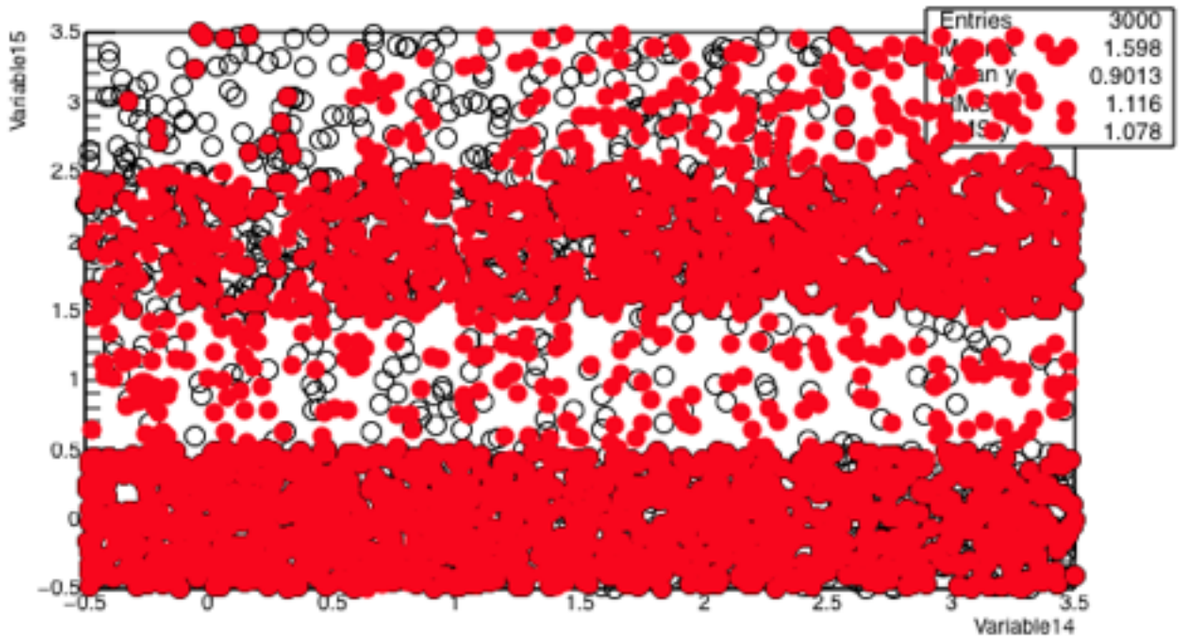


Exercise #2 Correlation Plots









Exercise #2 BDT Scores

- Surprisingly from looking at the data, a BDT can get decent separation