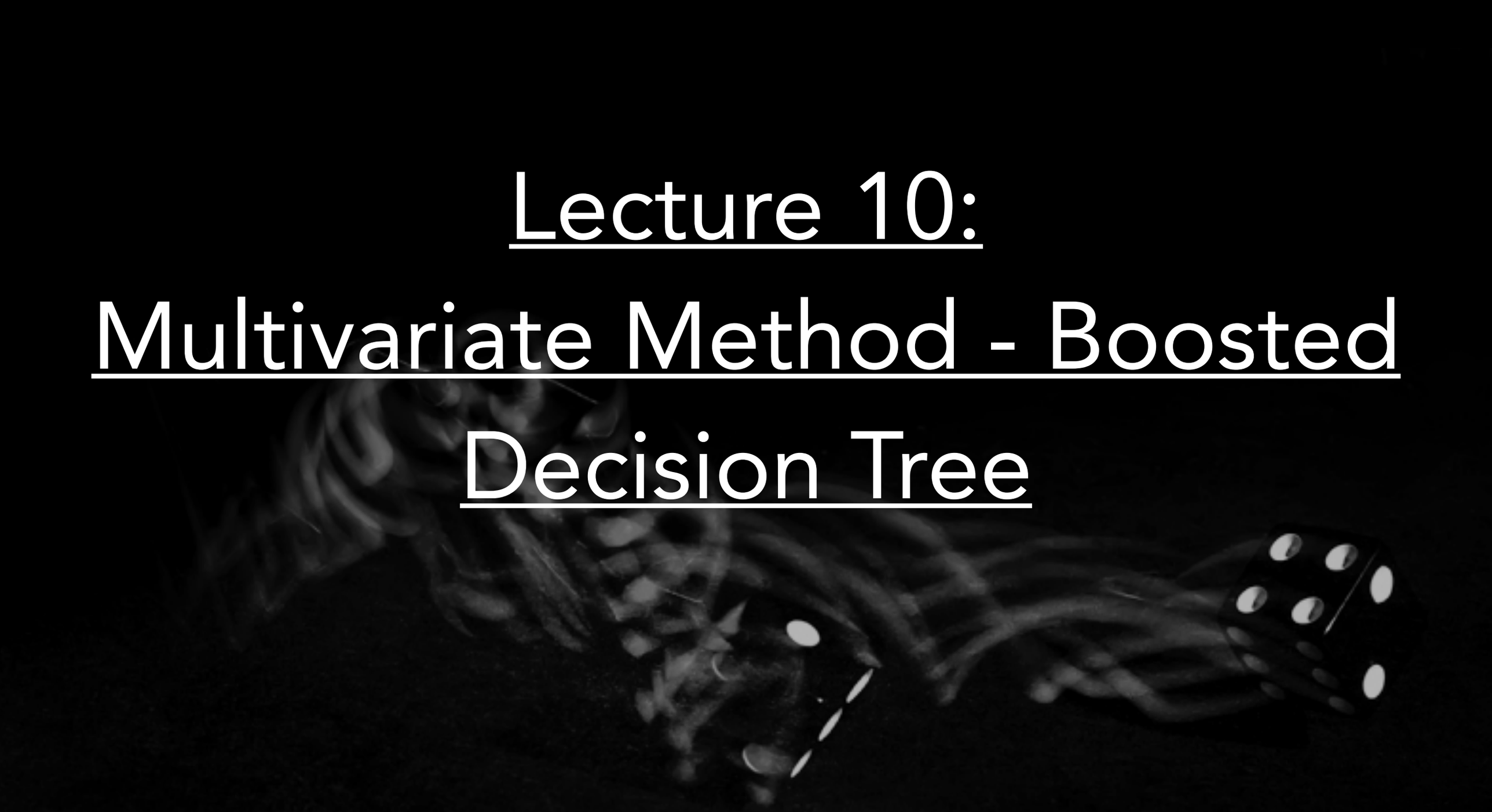
Lecture 10:
Multivariate Method - Boosted Decision Tree



D. Jason Koskinen
koskinen@nbi.ku.dk

Advanced Methods in Applied Statistics
Feb - Apr 2020

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

Photo by Howard Jackman

"Simple" Problems

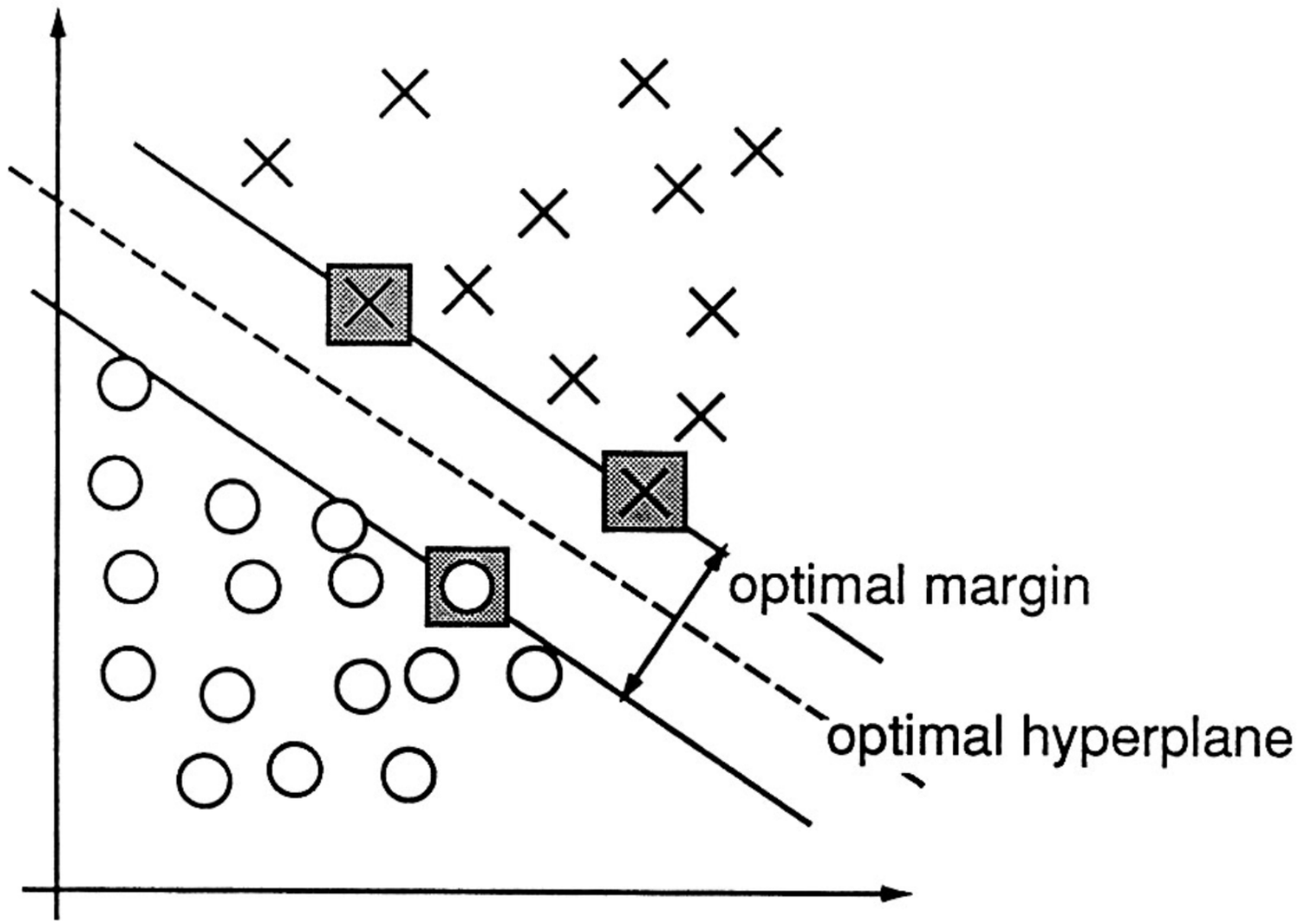
- Using likelihoods to separate background from signal is not always feasible
  - Likelihood may be too complicated for analytic or Monte Carlo evaluation
  - High dimensionality makes Monte Carlo computationally expensive
- Data sets which are linearly separable in variables, e.g. between signal and 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

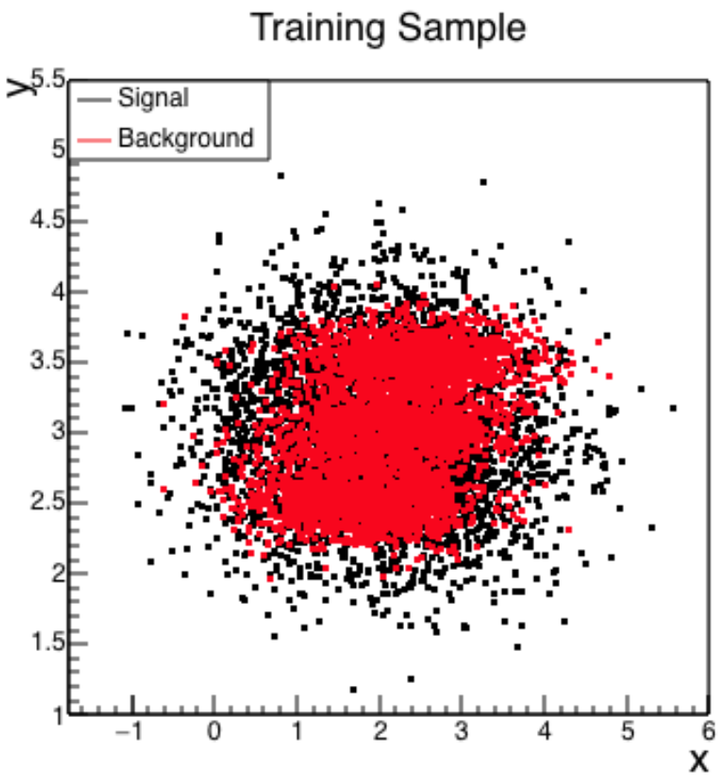
- Depending on the kernel, the SVM maps the data into a higher or alternate dimension space, and makes classifications via hyperplanes

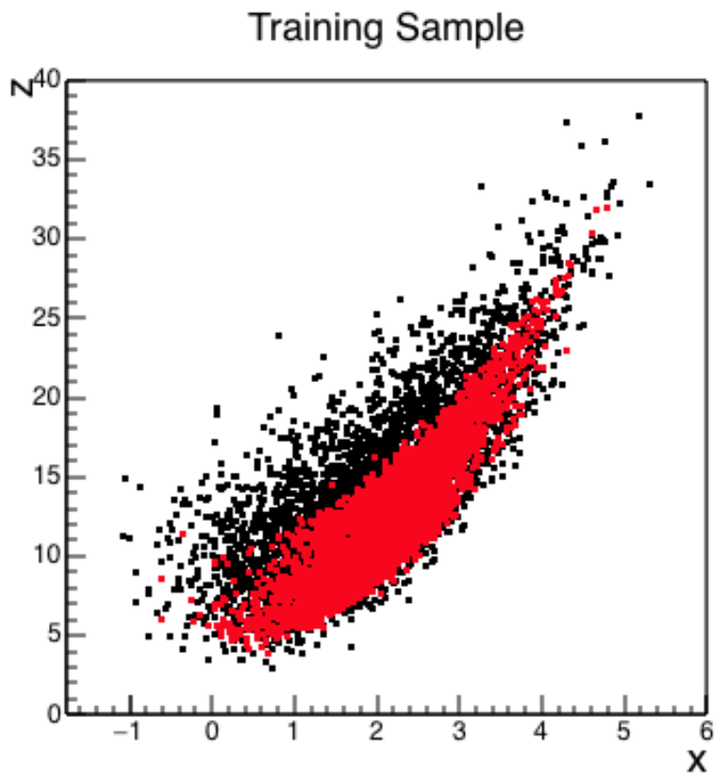


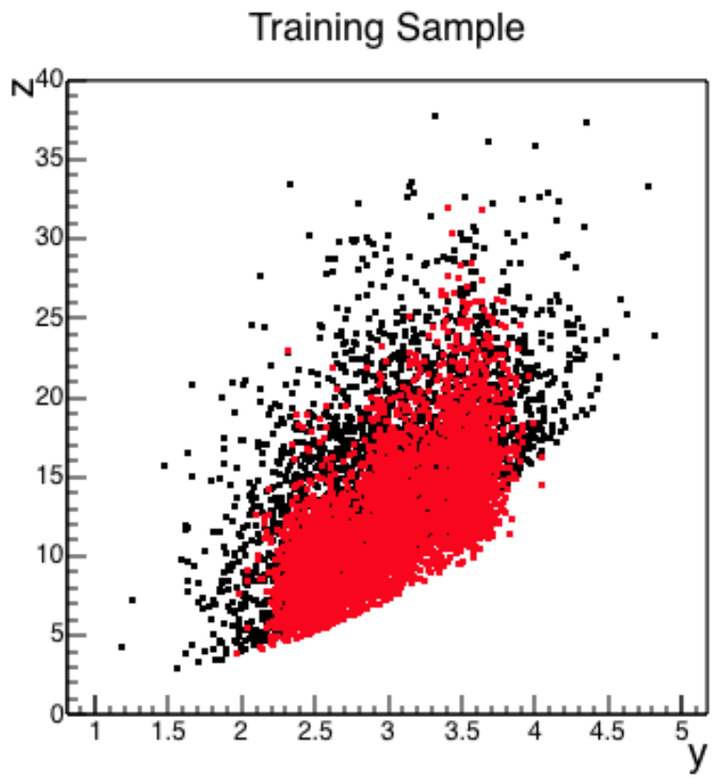
*"Support Vector Networks", Cortes & Vapnik

Training Samples

- Below are events in only x, y, and z for some class of signal and background, which are not obviously easy to separate via straight cuts
  - Machine algorithms can ‘see’ in higher dimensions very quickly
  - This semi-simple example with a boosted decision tree can get a true positive rate of 88% at 1% false positive rate





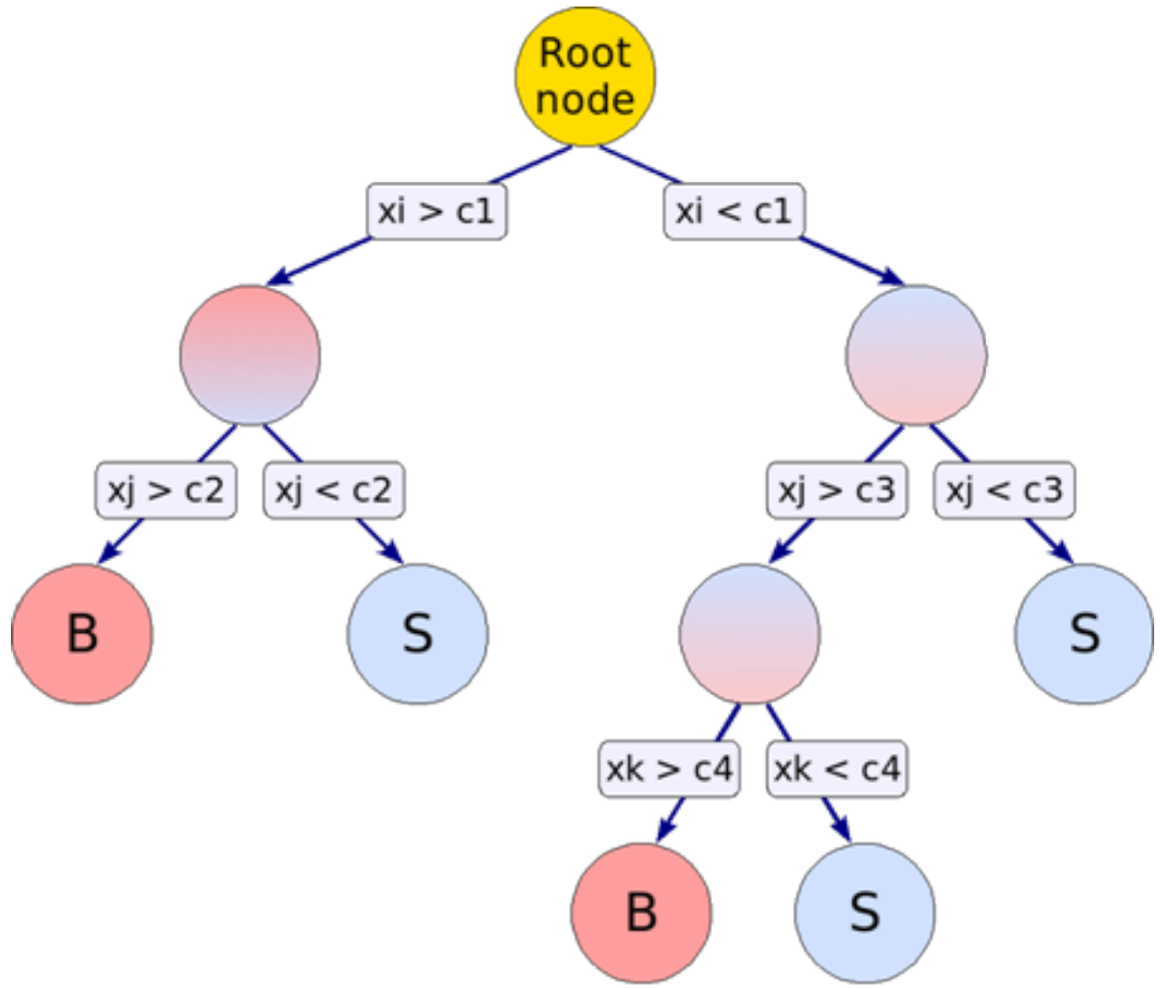


Clarity

- Statistical terms can be different depending on the research field
- A standard in Machine Learning is the terms associated with a confusion matrix (https://en.wikipedia.org/wiki/Confusion_matrix)

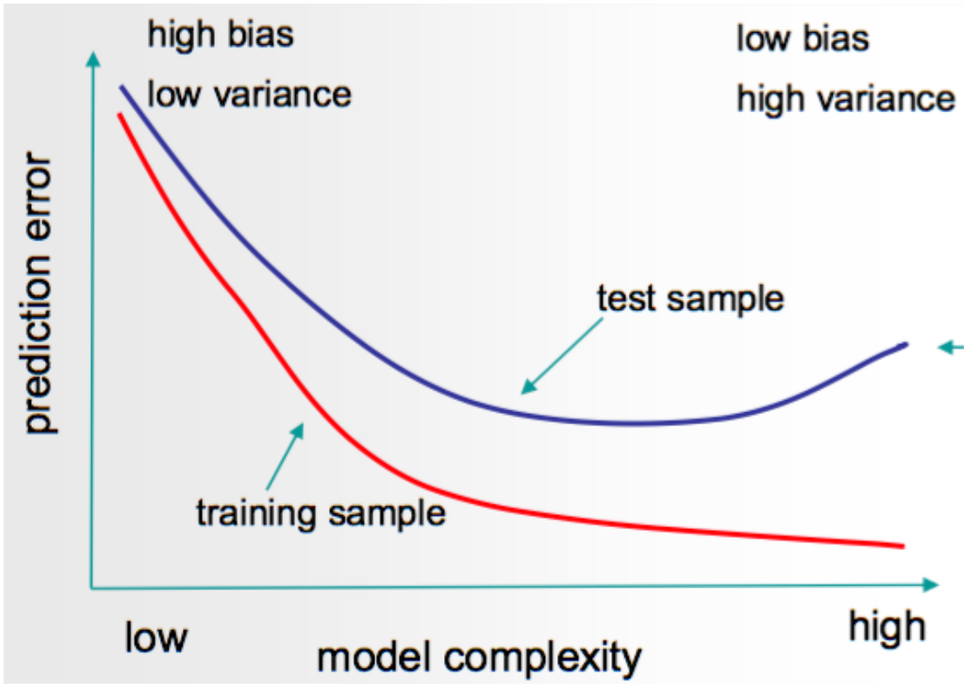
Decision Tree

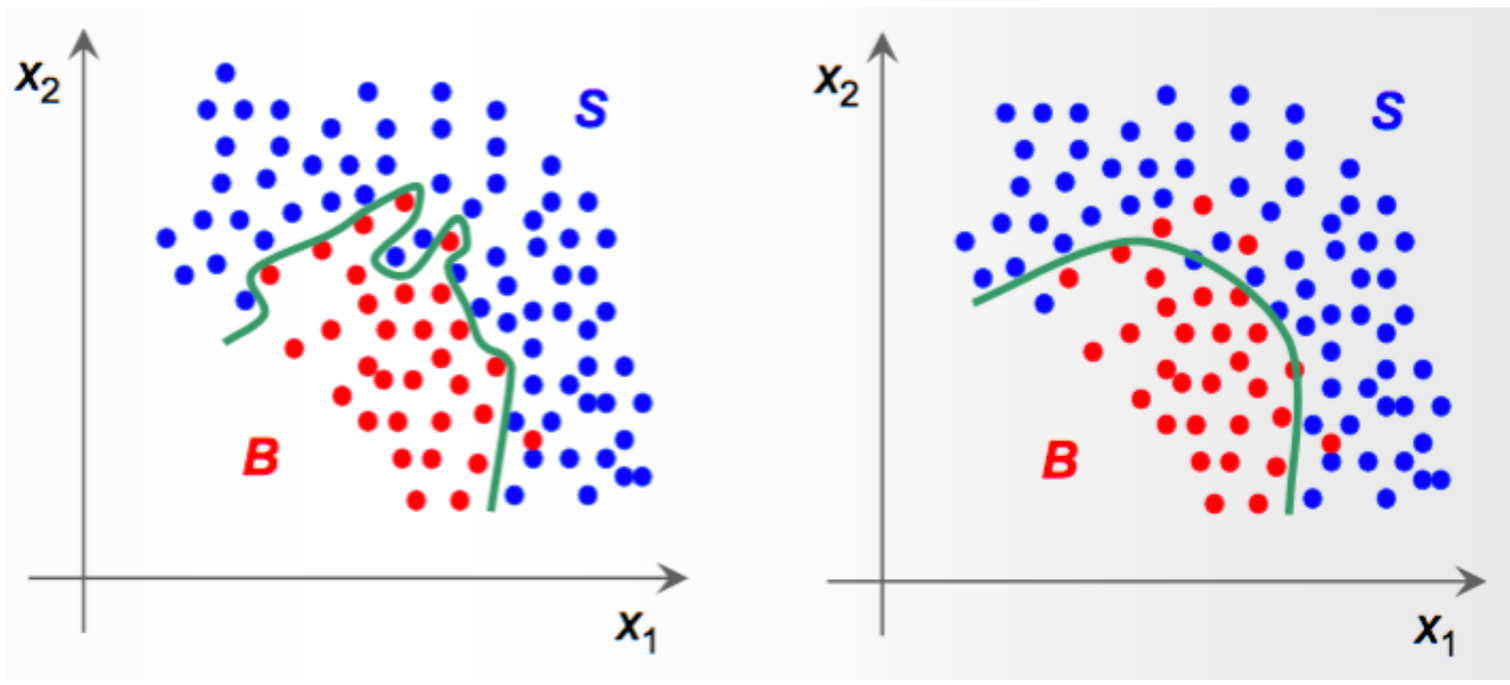
- Decision Tree: Sequential application of cuts splits the data into nodes, where the final nodes (leaves) classify an event as signal or background
  - Easy to visualize and interpret
  - Resistant to outliers in data
- Disadvantage is that statistical fluctuations in training create instability



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

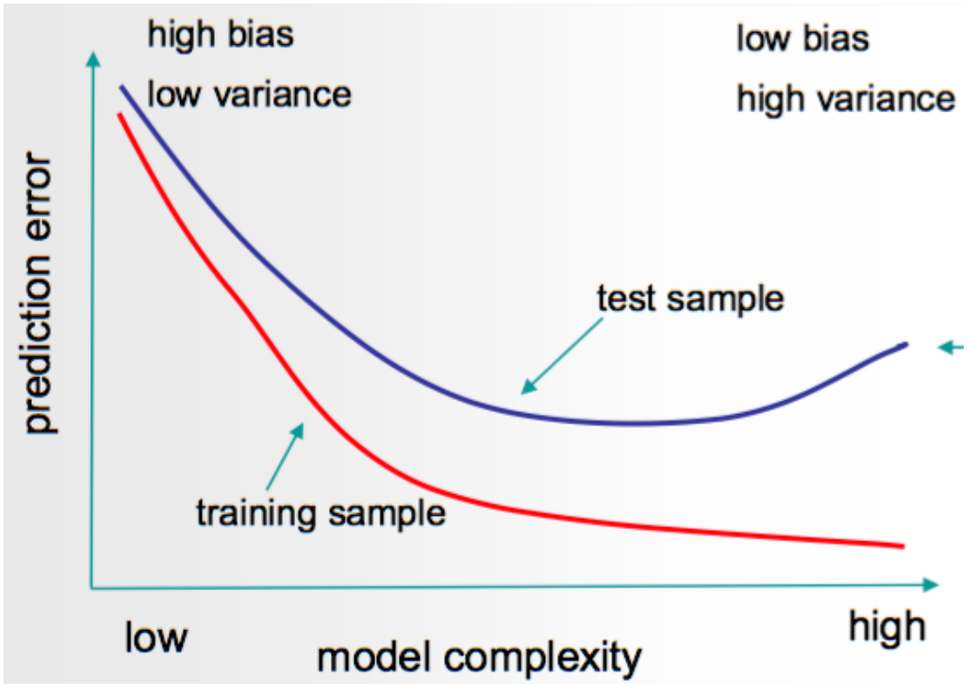


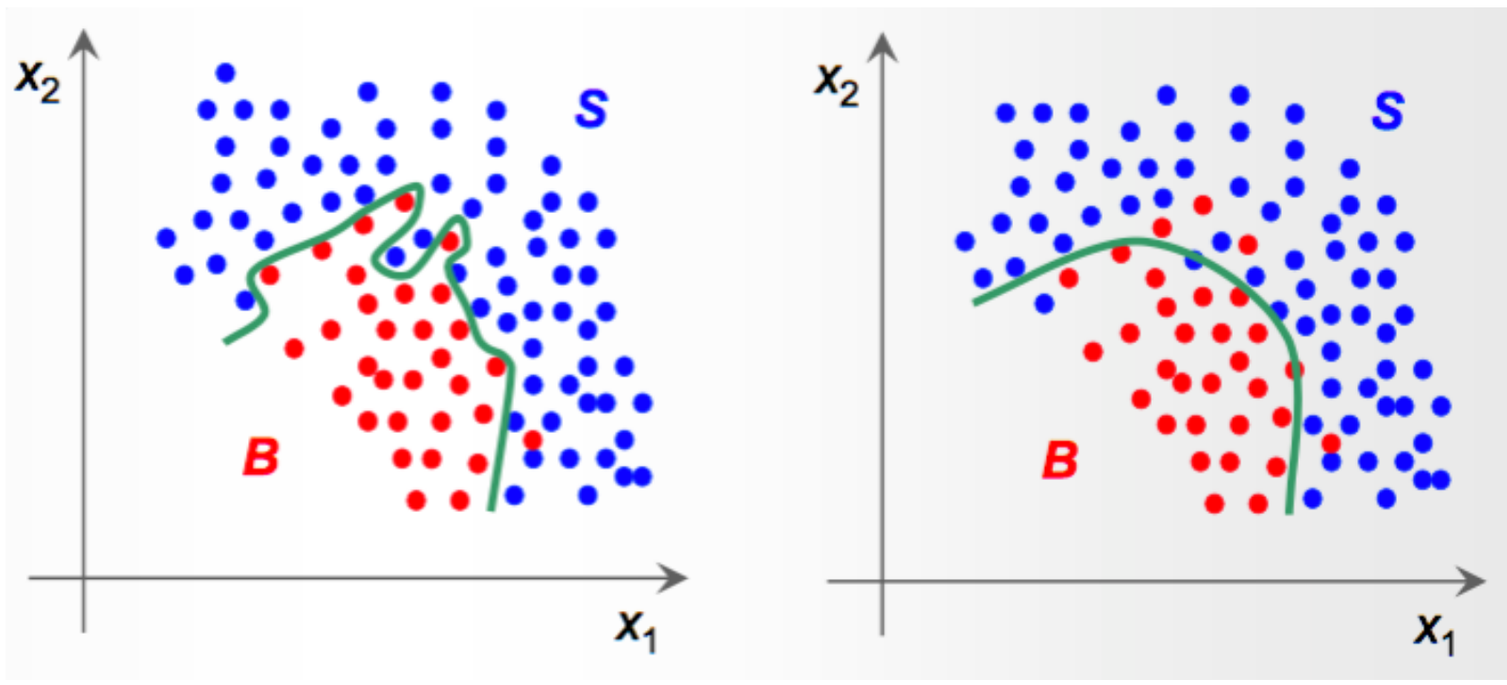


*H. Voss (MPIK)

Overtraining

- Just because the training uses lots of data, doesn't mean the algorithm is overtrained. It's the balance between statistical fluctuations in the training sample AND the complexity of the model that drive 'overtraining'.

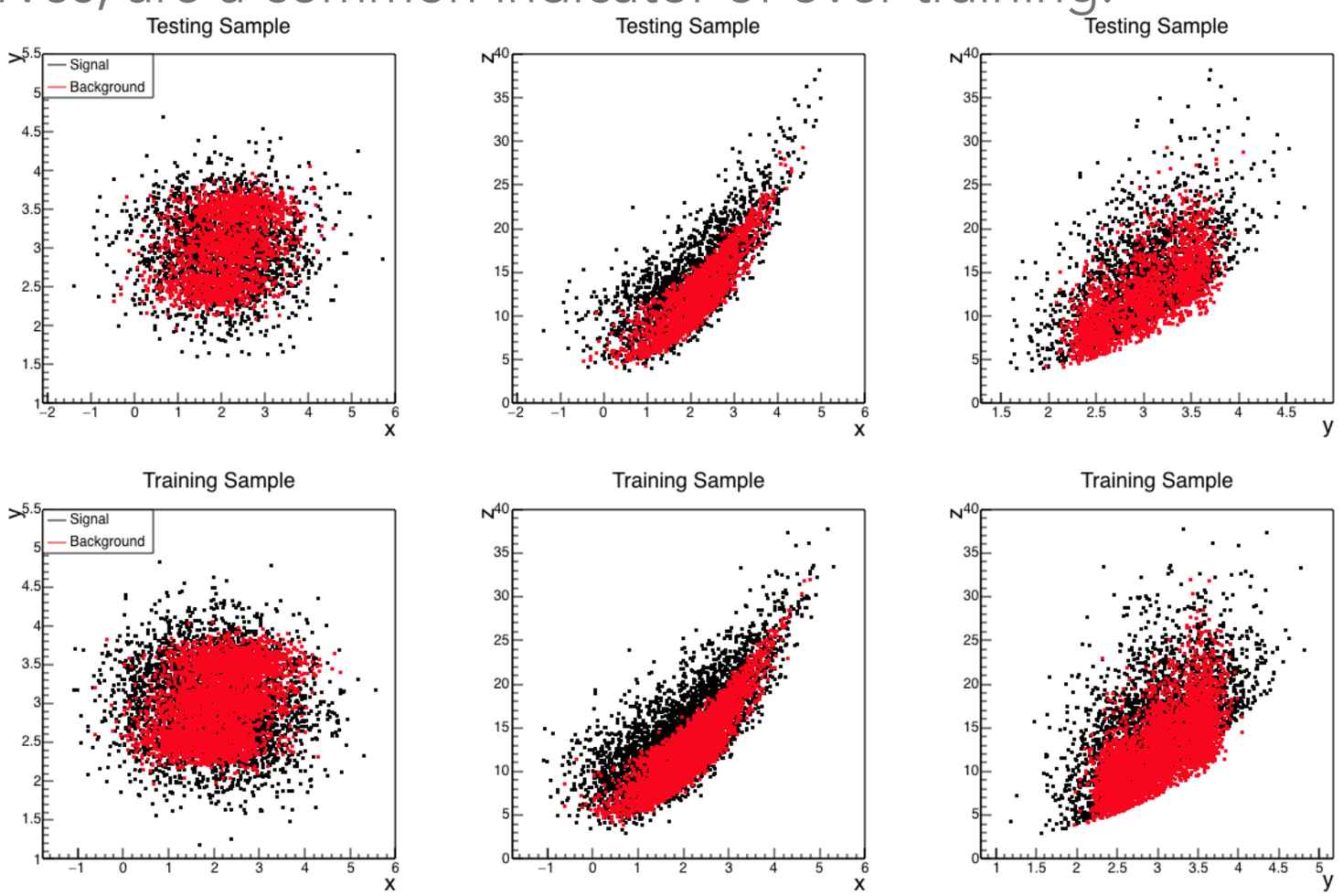




*H. Voss (MPIK)

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.



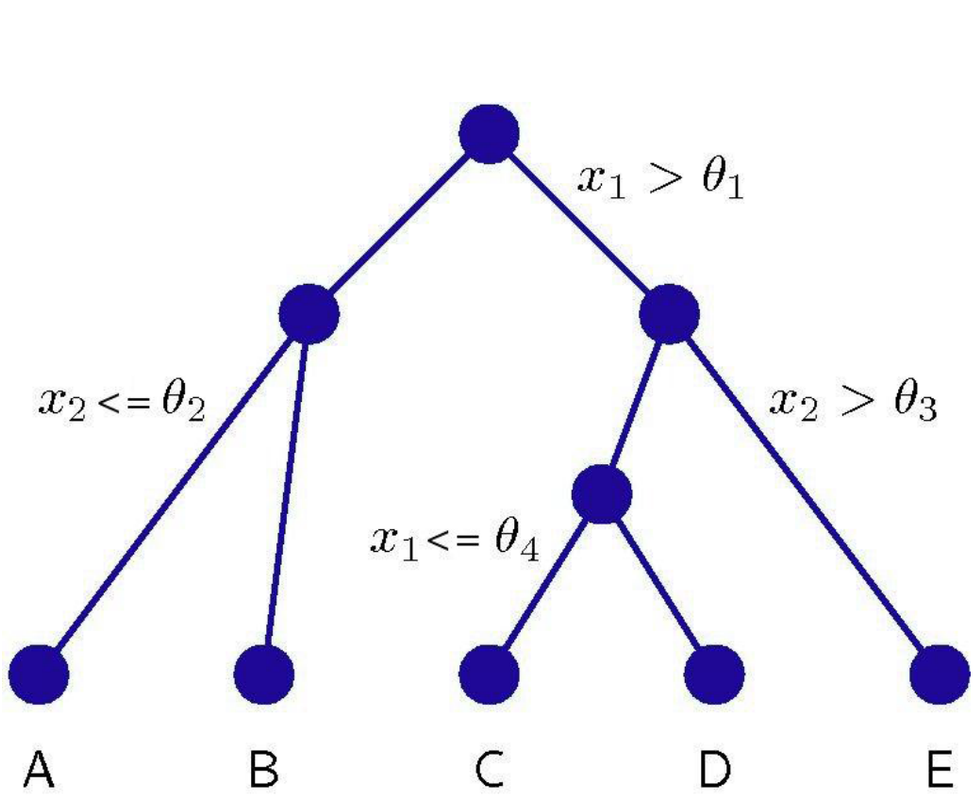
*n.b. just because the training uses lots of data, doesn't mean the algorithm is overtrained. It's the balance between statistical fluctuations in the training sample **AND** the complexity of the model that drive 'overtraining'.

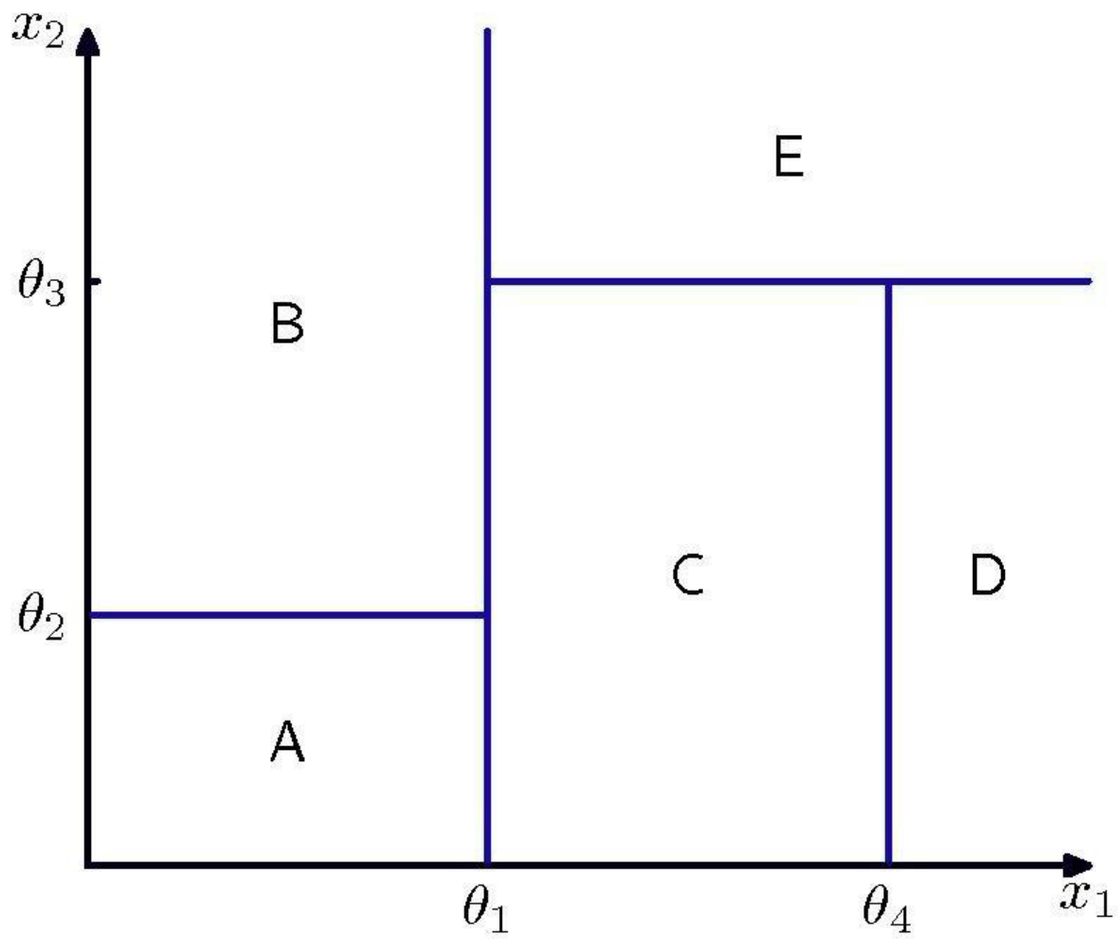
Creating the Decision Tree

- 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 x_1 and x_2 classified as signal/background regions by the decision tree



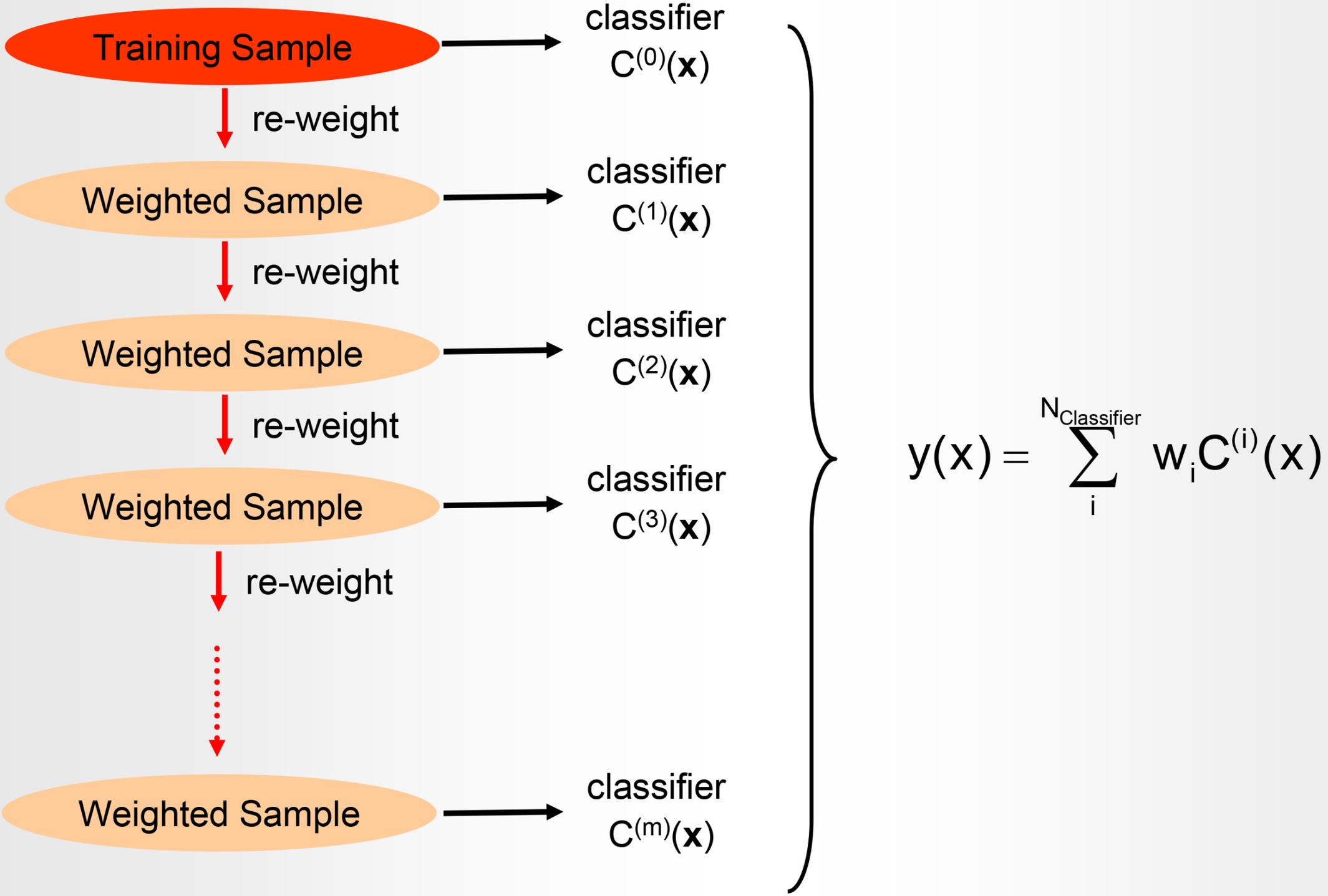


*J. Therhaag

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 helps, due to 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 to avoid identical decisions with multiple iterations is to use Boosting

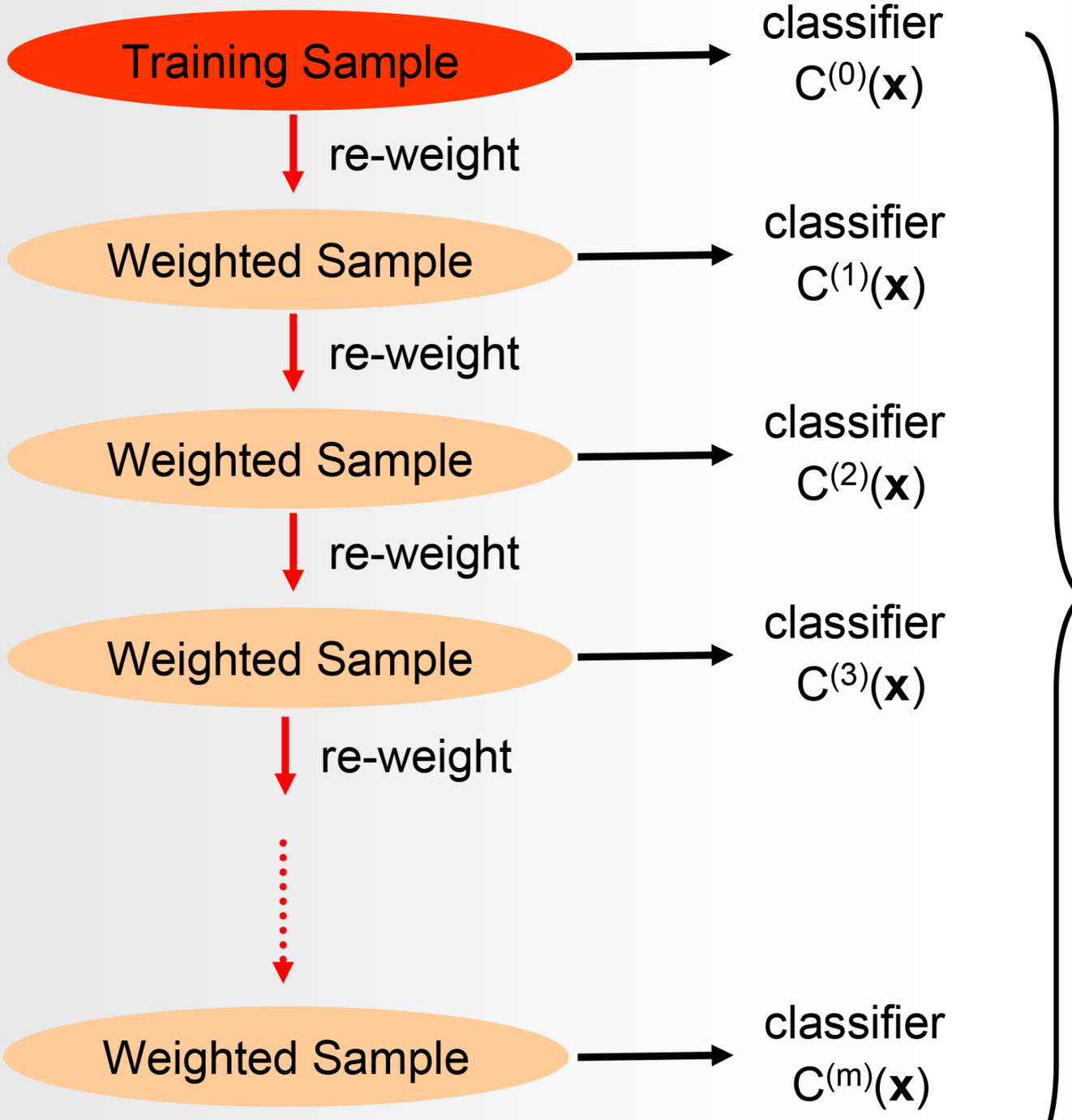
Boosting


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
$C^{(m)}(\mathbf{x})$
$y(x) = \sum_{i}^{N_{Classifier}} w_i C^{(i)}(x)$

Boosting Algorithm

- While all boosted decision trees have the same principal, the algorithms that determine the 'boosting', splitting and pruning of the leaves, etc. are different
- Gradient boosting minimizes the residual error from the prediction/classification of the previous 'tree', and uses gradient information to inform the new tree(s)
- Adaptive boosting iteratively modifies training events with new importance 'weights' in order to improve learning

Adaptive Boosting (AdaBoost)



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

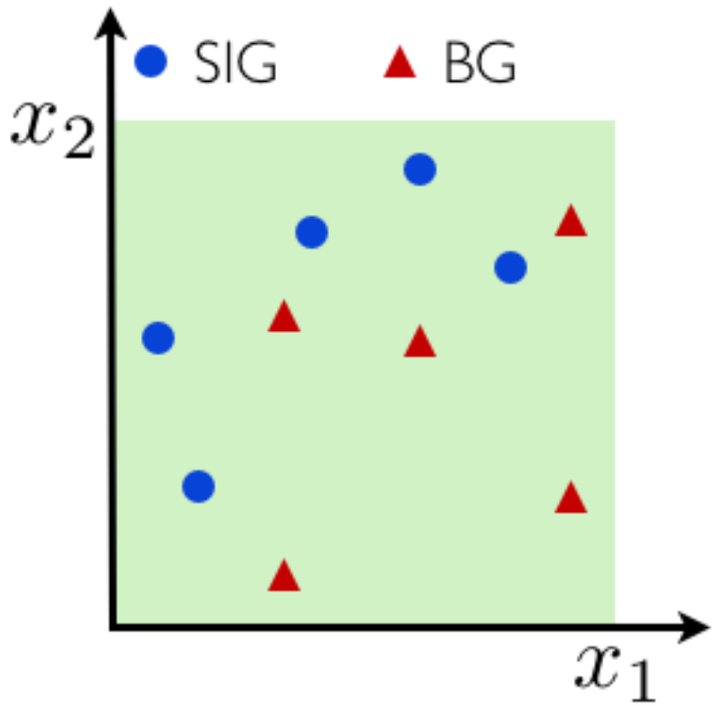
$$\frac{1-f_{err}}{f_{err}} \quad \text{with :}$$

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

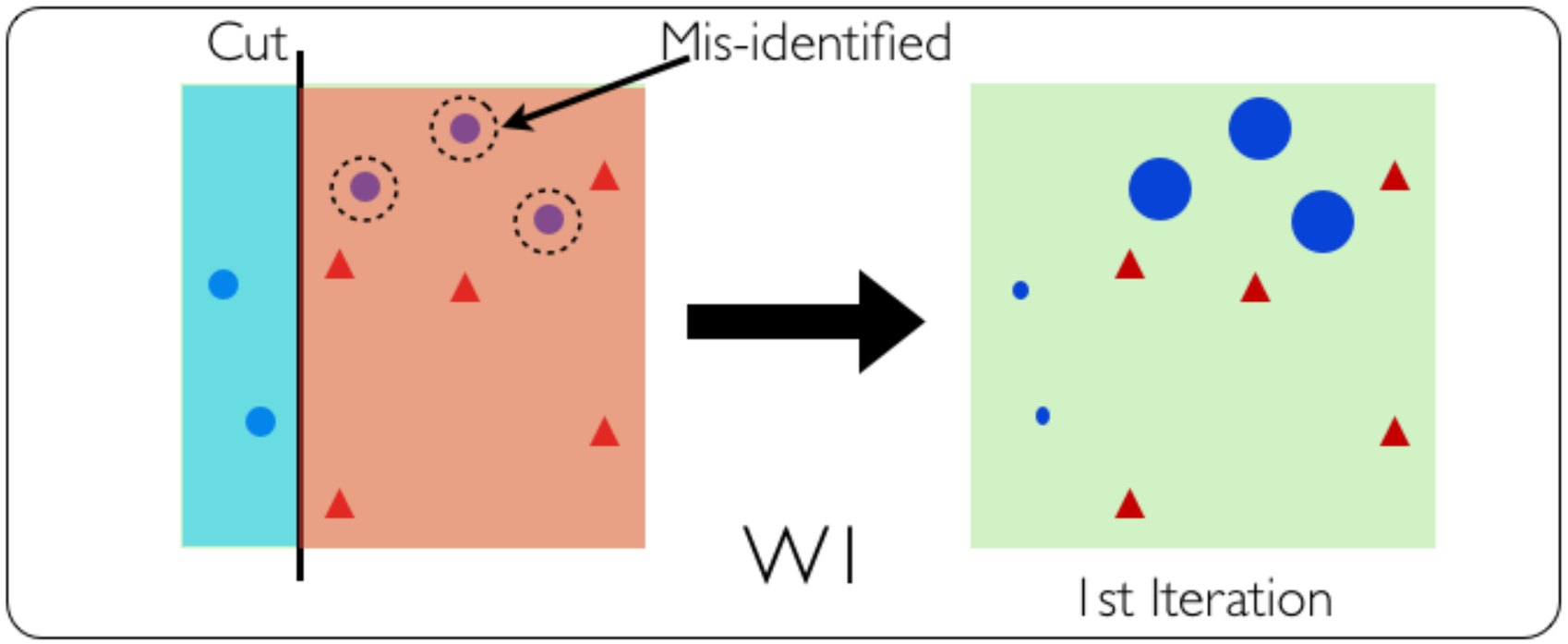
- AdaBoost weights the classifiers also using the error rate of the individual classifier according to:

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

Adaptive Boosting Visually



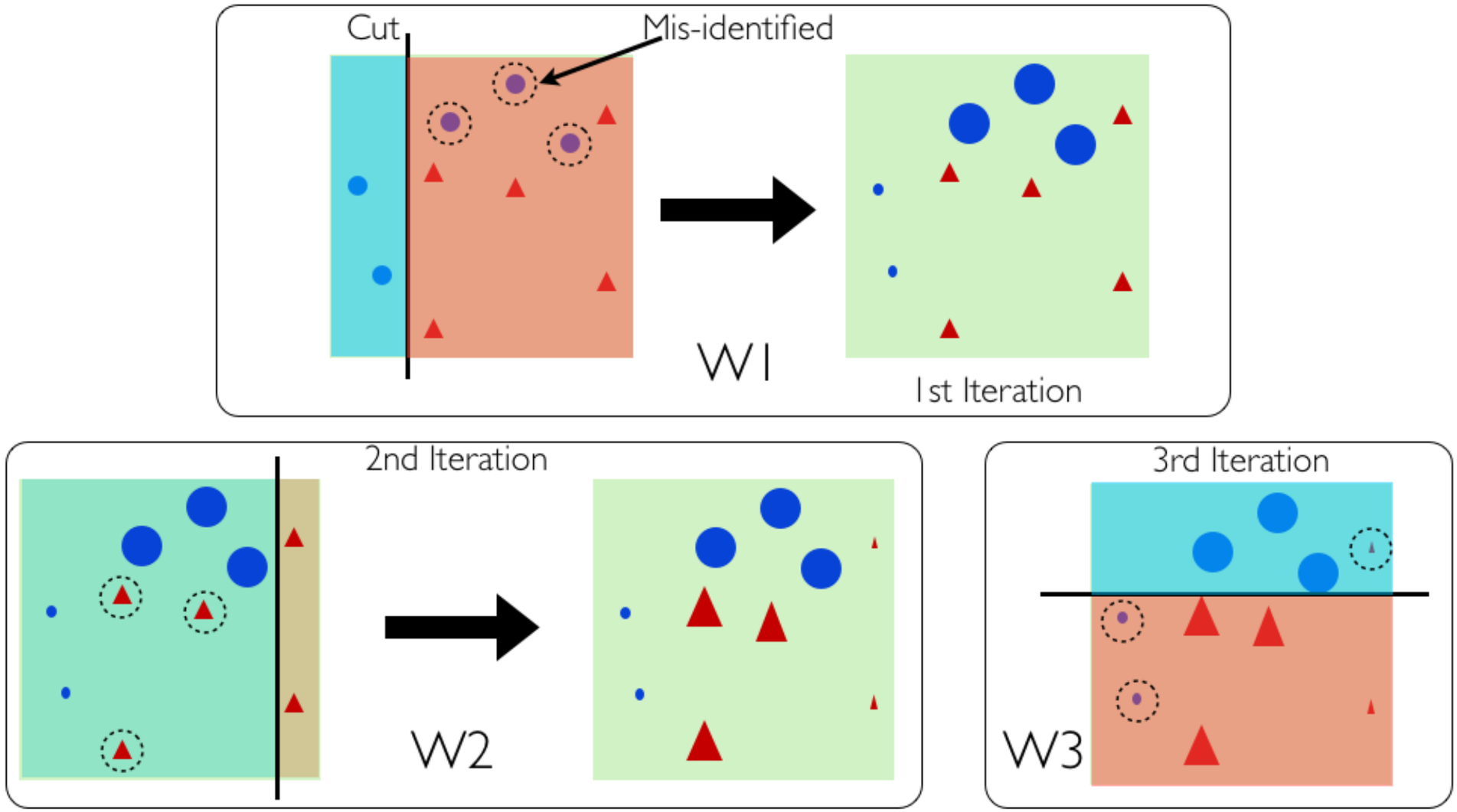
<Simple 2-D toy example>
Hypothesis : Straight Cuts
Multiple Iterations
Boosting : Give different weights
when (mis)classified



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

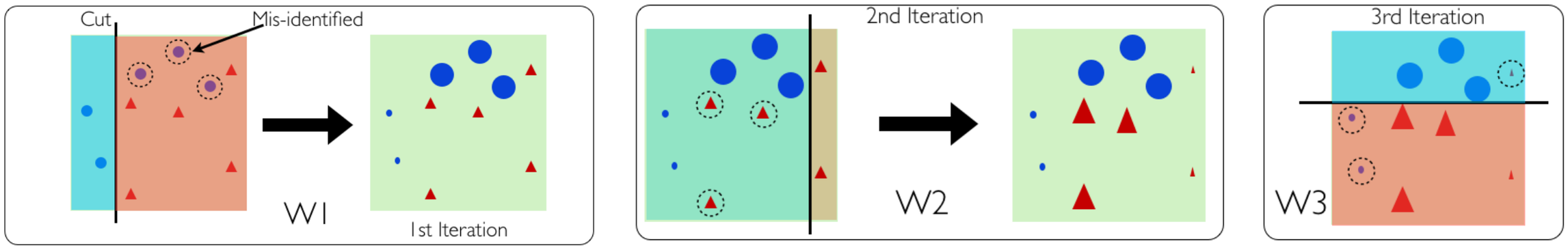
AdaBoost Boosted Decision Trees

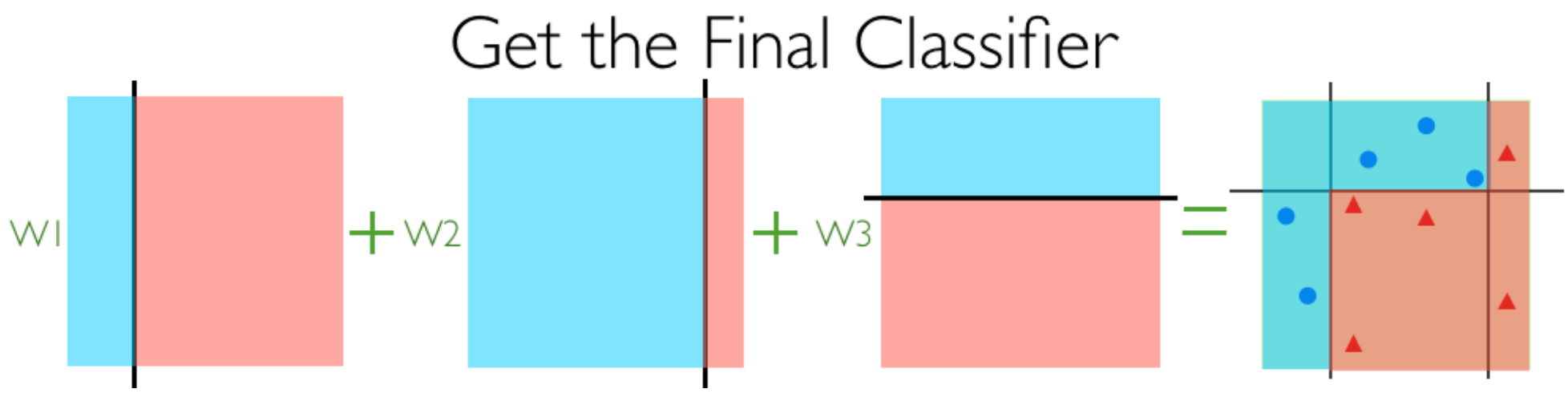
- Past the first one, each iterative boosted decision tree (classifier) is trained on the ‘same’ events. But now, the events have weights according to whether they were previously wrongly classified. Also, the regions have weights (W1, W2, W3 in the example below) corresponding to the classification error.



AdaBoost Boosted Decision Trees

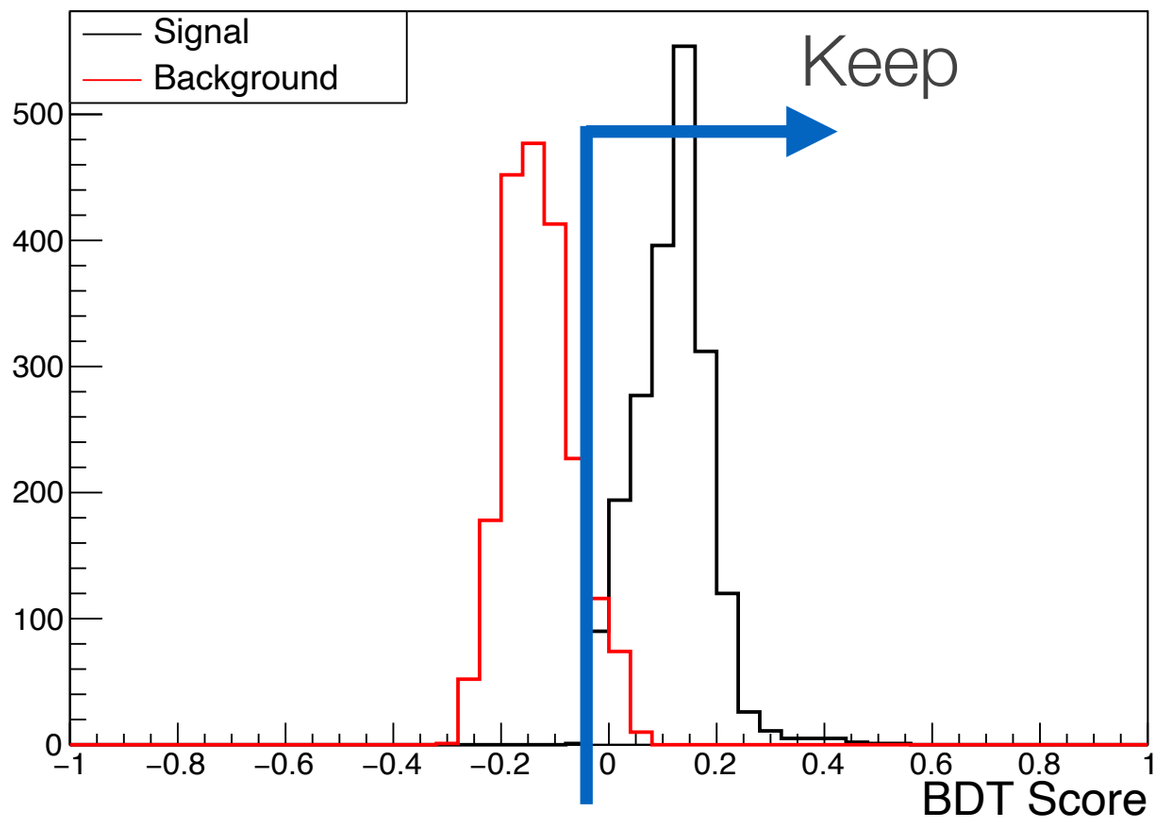
- The combined classifier is the weighted average from all trees for the different regions
- Works very well "out-of-the-box"





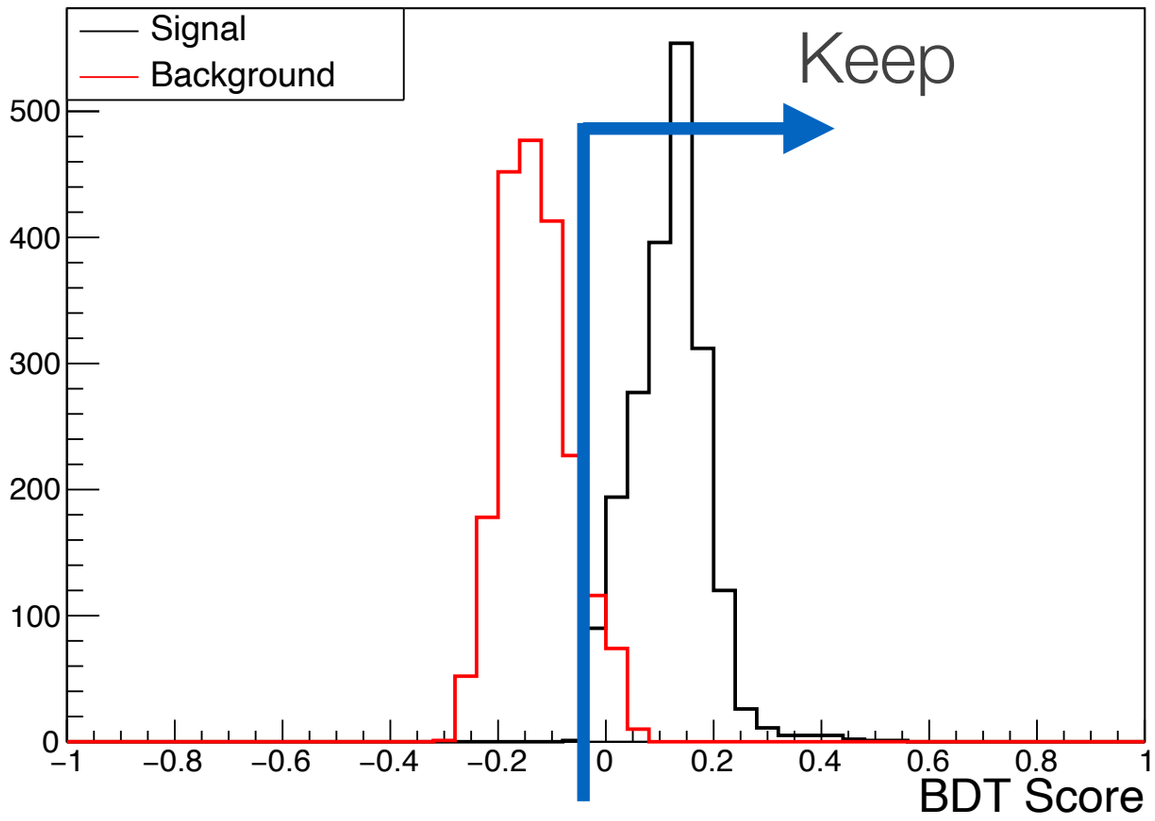
Boosted Decision Tree Classifier

- After training, and hopefully testing, the BDT can generate a score when run over new data that allows signal/ background separation
  - Place a cut at some value of the test statistic (here it is the BDT score) to get desired precision and true positive rate



Boosted Decision Tree Classifier

- Different software packages and algorithms will return different ‘test statistics’ for the classification
  - From AdaBoost, we get the ‘decision score’ as the test statistic. More negative values are background.
  - XGBoost gives a probability for an event to be background or signal, which is the test statistic.



We are using the BDT as a classifier and want a decision about whether a **new** data event is more similar to class-A or class-B, e.g. signal or background. In AdaBoost we use a “BDT Score” which is here the BDT decision score.

BDT Comments

- It is common to throw an absurd number of variables into a BDT and have the software signify the variables of importance. 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, size of training sample, 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.

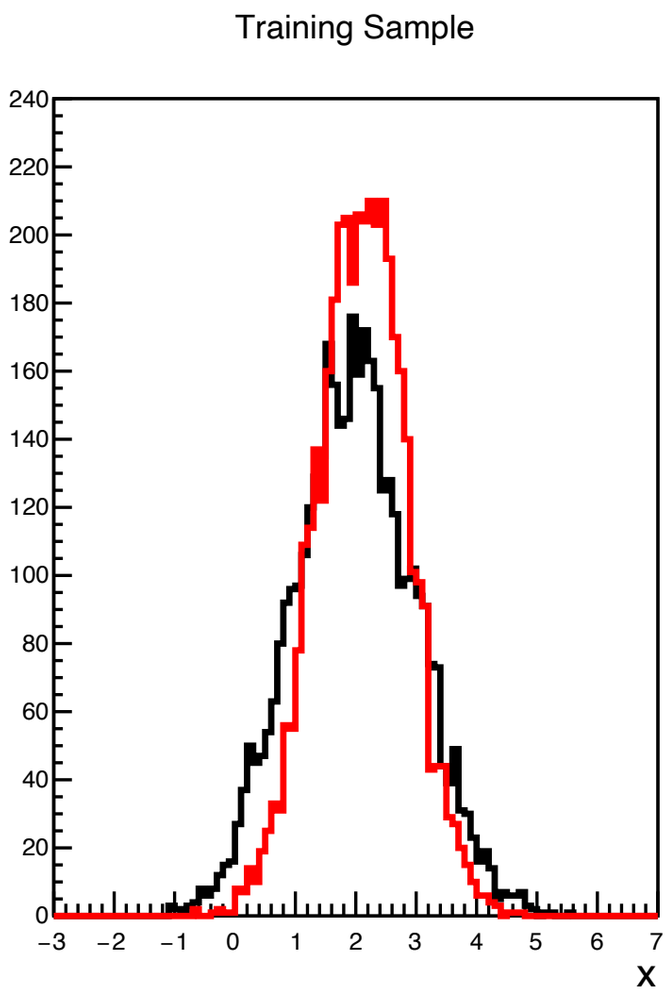
Exercise #1

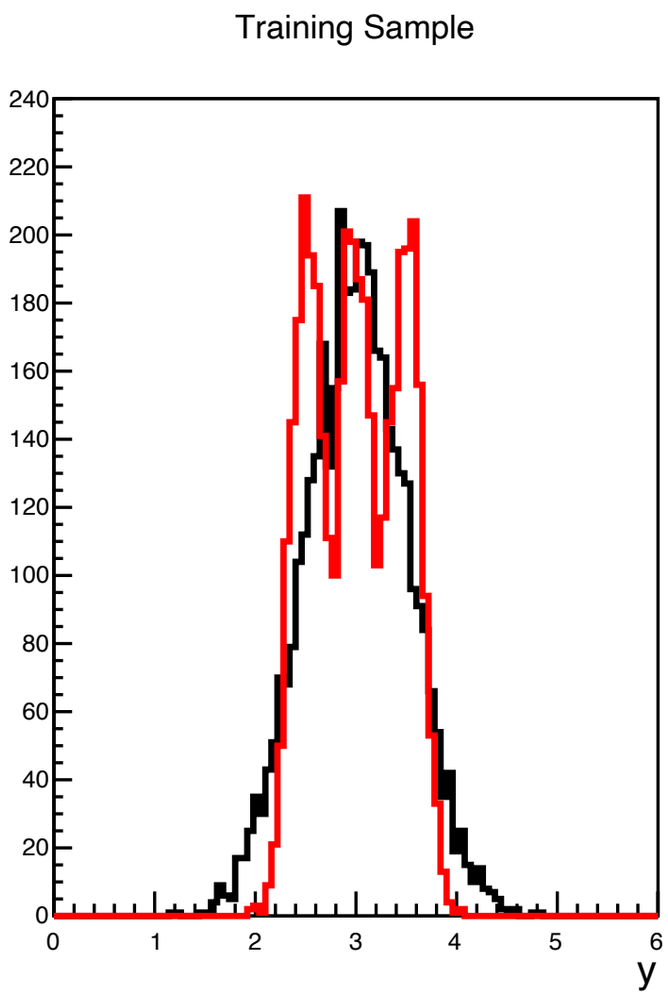
- On the class webpage there are 4 files which include signal and background for training and testing of a learning algorithm (nominally a boosted decision tree)
- The data is generated as an unknown function of three variables

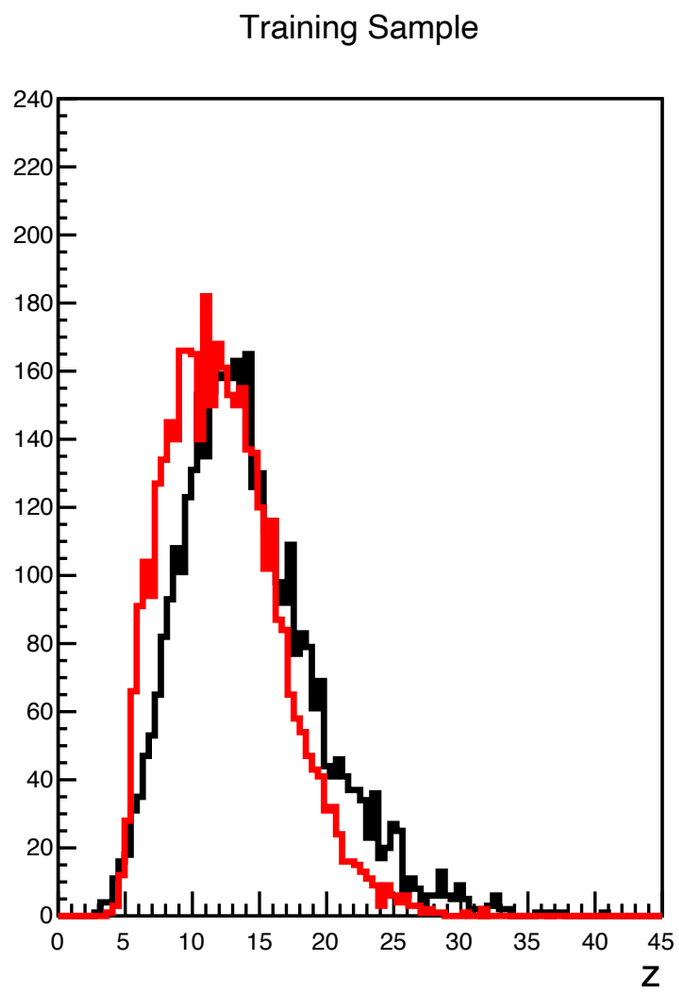
- Plot the three variables for the signal and background samples for training:
  - 1D histograms, one for each variable
  - 2D graphs of the x vs. y, x vs. z, and y vs. z
- After training the BDT, plot the BDT decision score for the test sample separated by color for the signal and background

Exercise #1 Simple Plots

- Not a whole lot of separation going on here

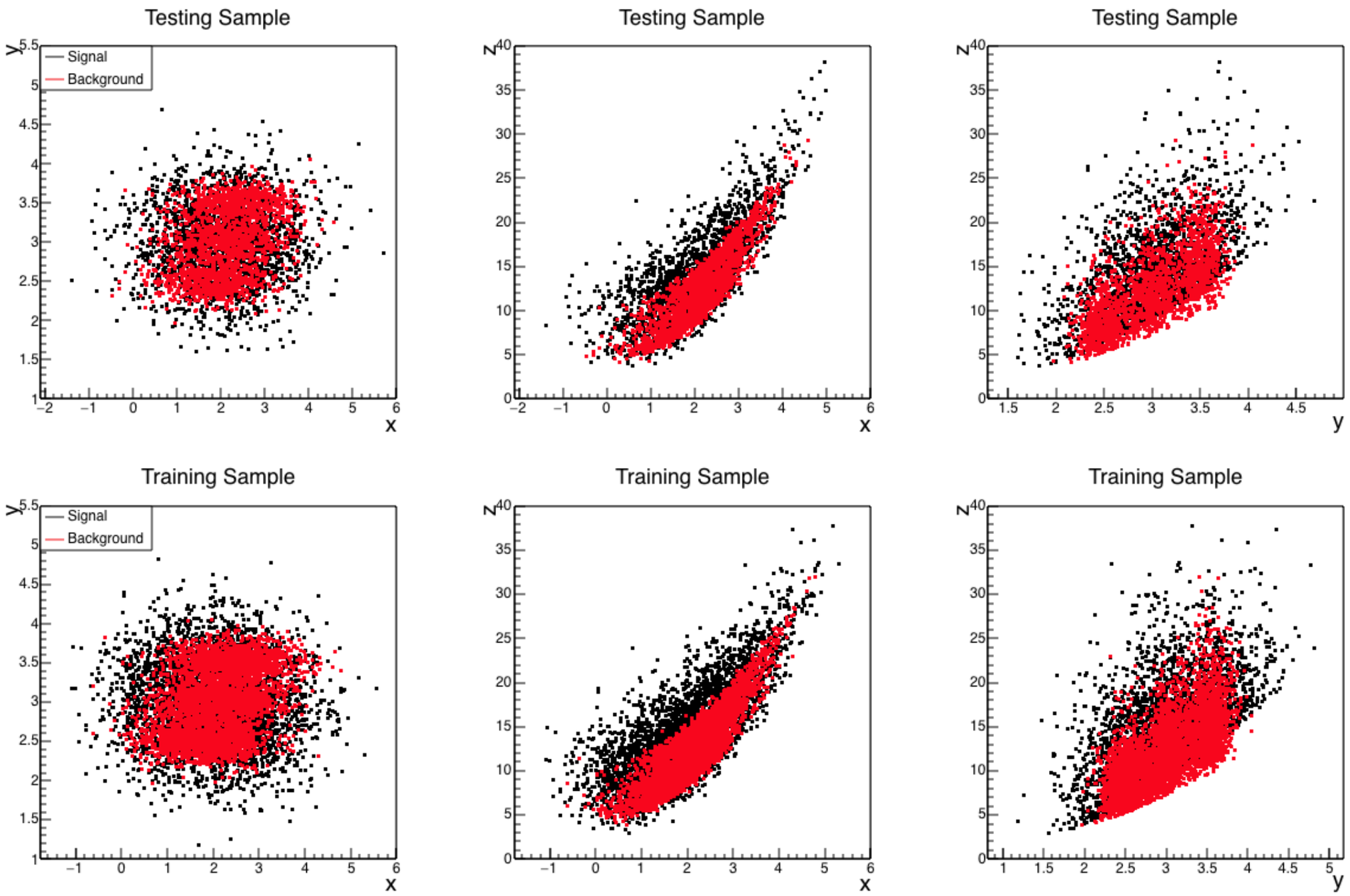




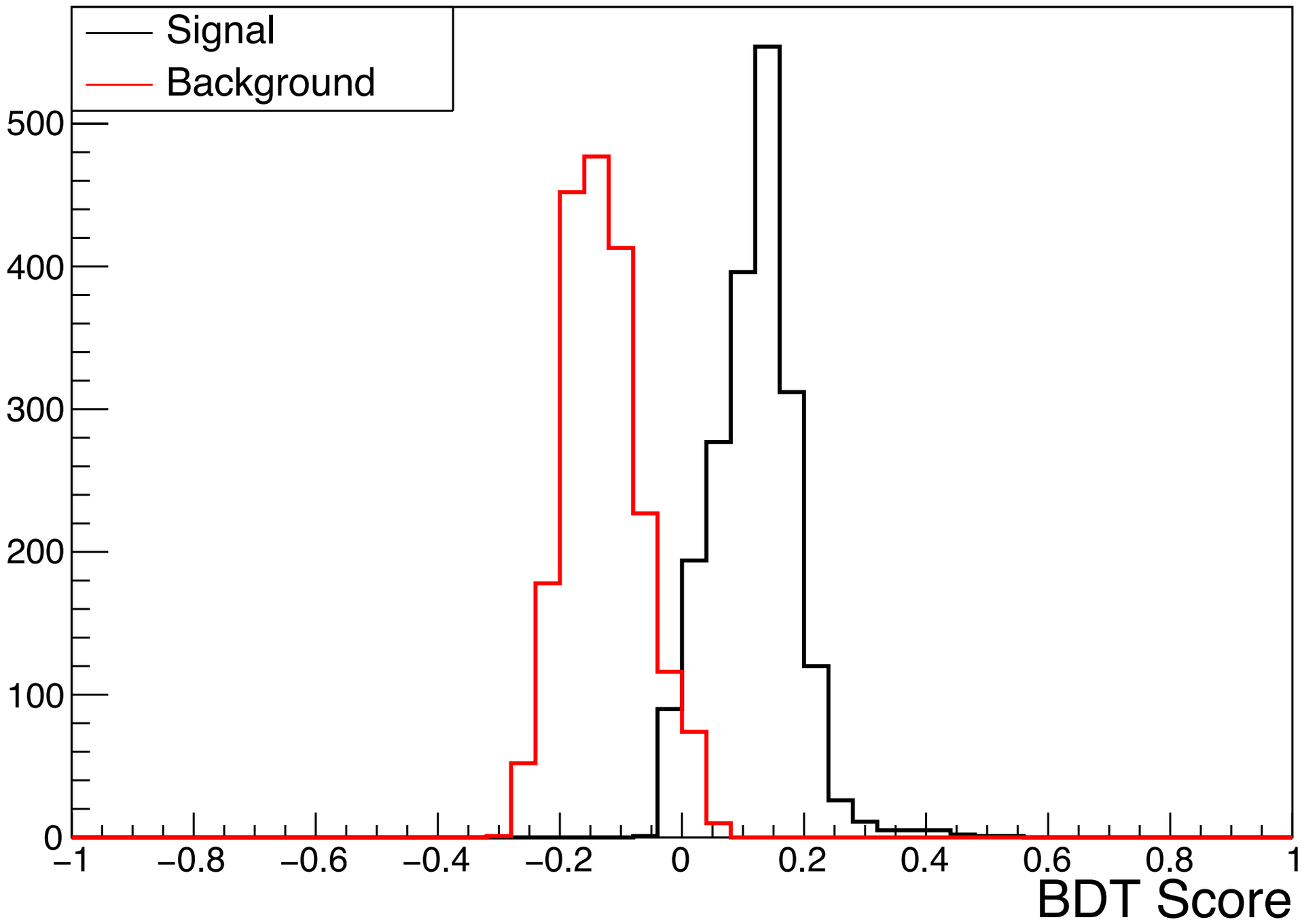


Exercise #1 in 2D

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



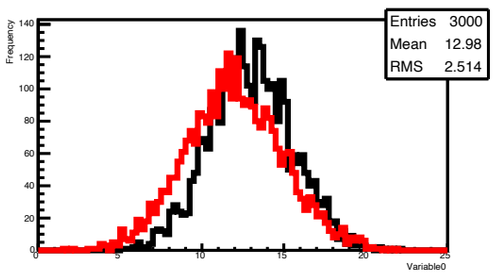
Exercise #1 BDT Score Result

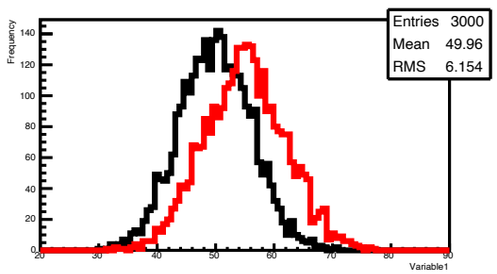


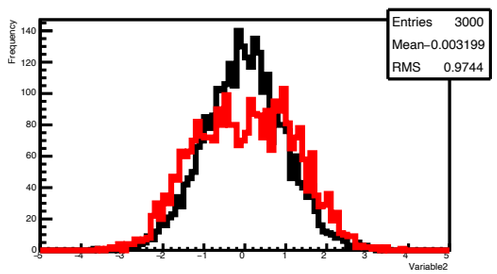
Exercise #2

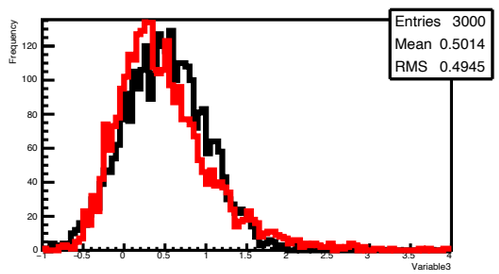
- Now with 16 variables using the single file on the webpage
- Separate the data from the file into a training and testing sample and repeat
  - Do not use any training data as a test sample
- Train a BDT (or some other machine learning algorithm) and plot the output scores
- Are any variables essentially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

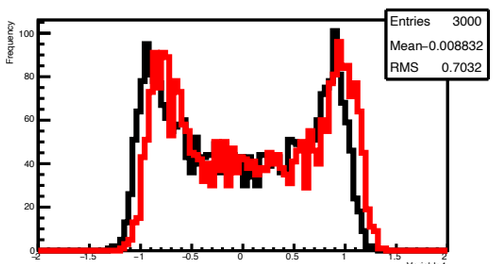
1D Histograms of the 16 Variables

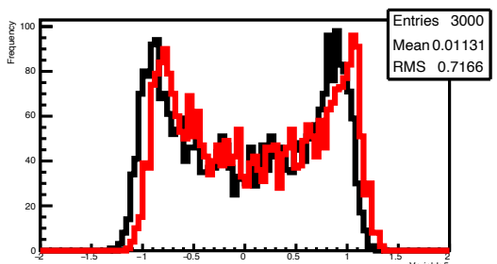


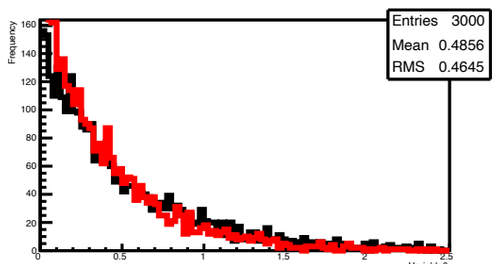


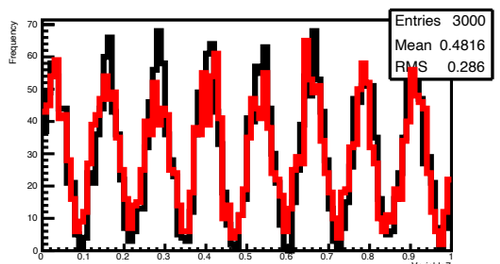


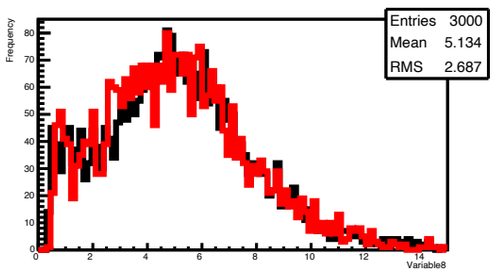


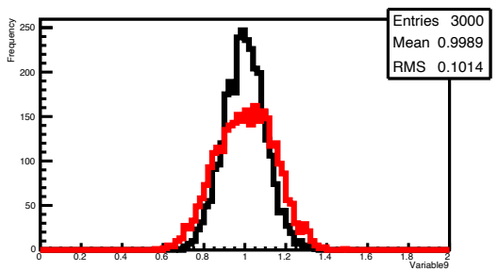


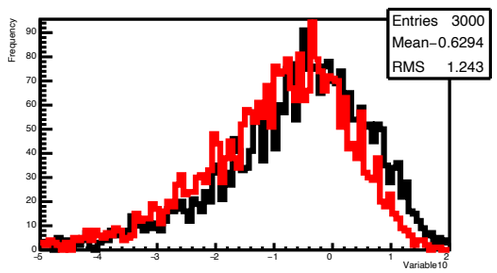


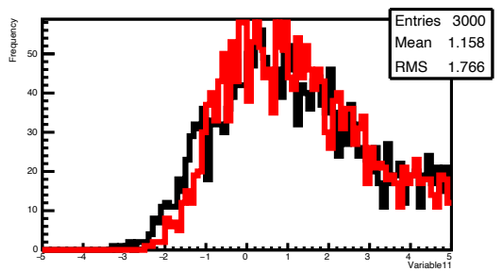


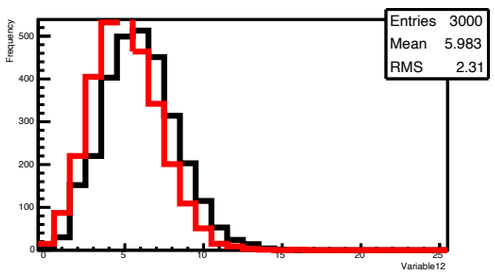


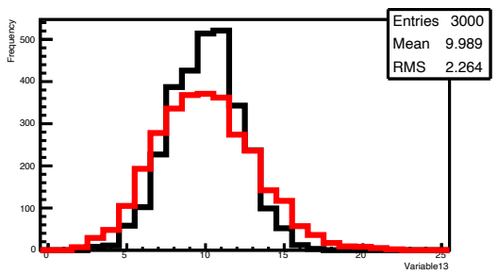


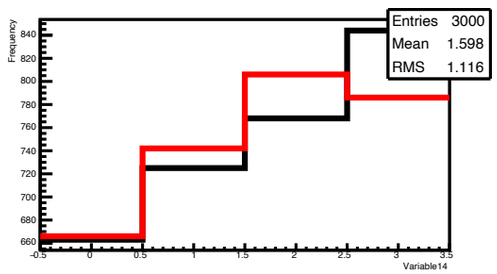


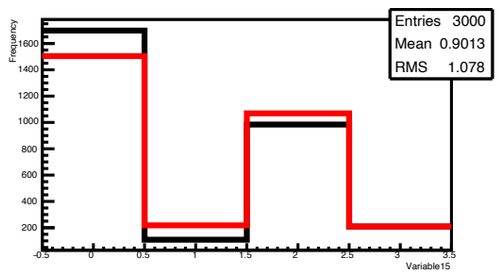




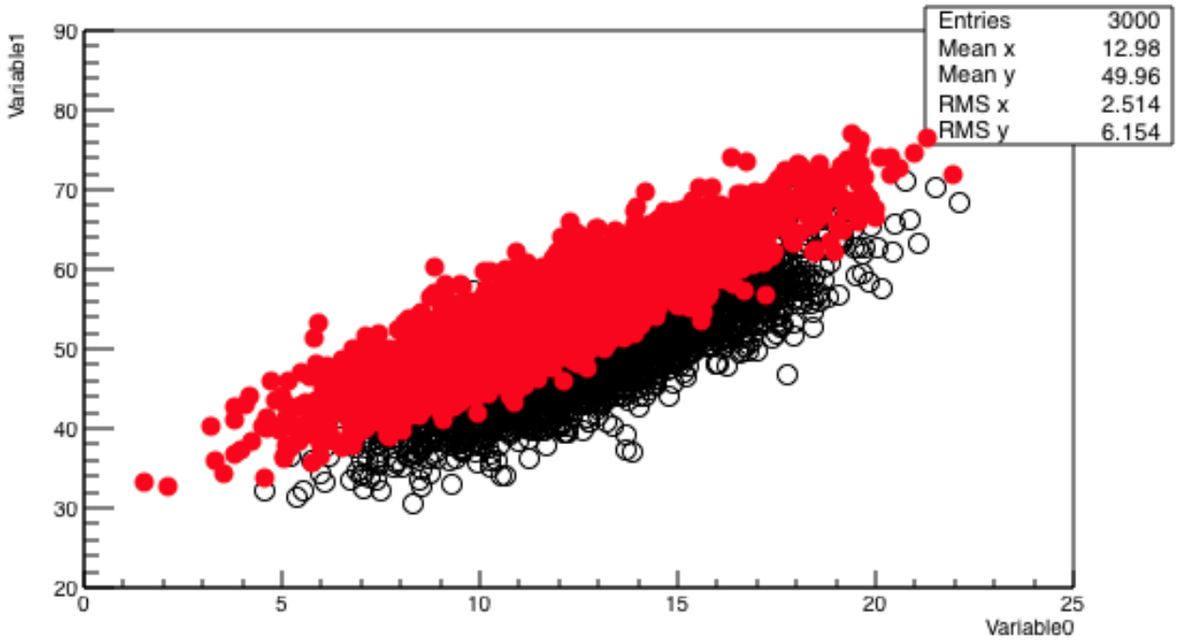


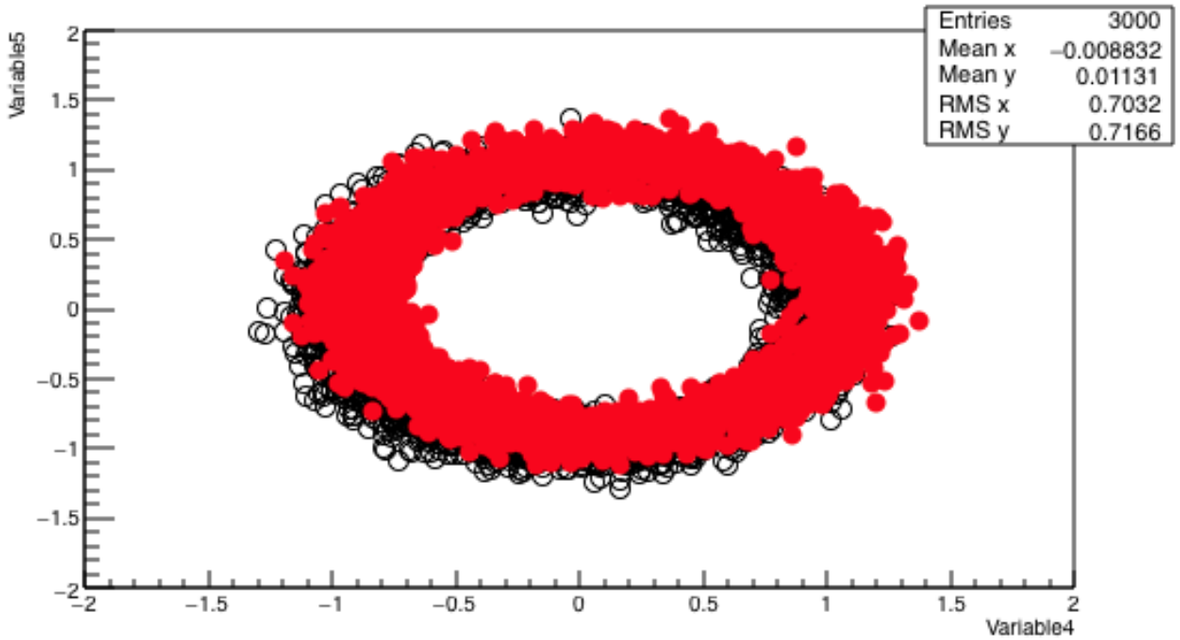


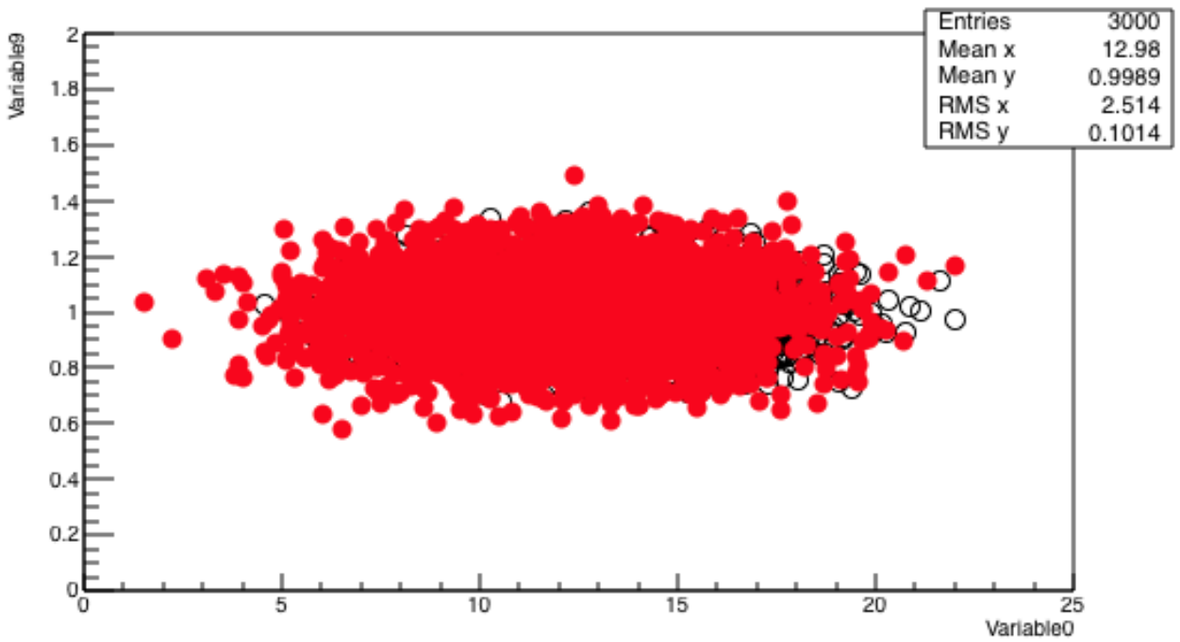


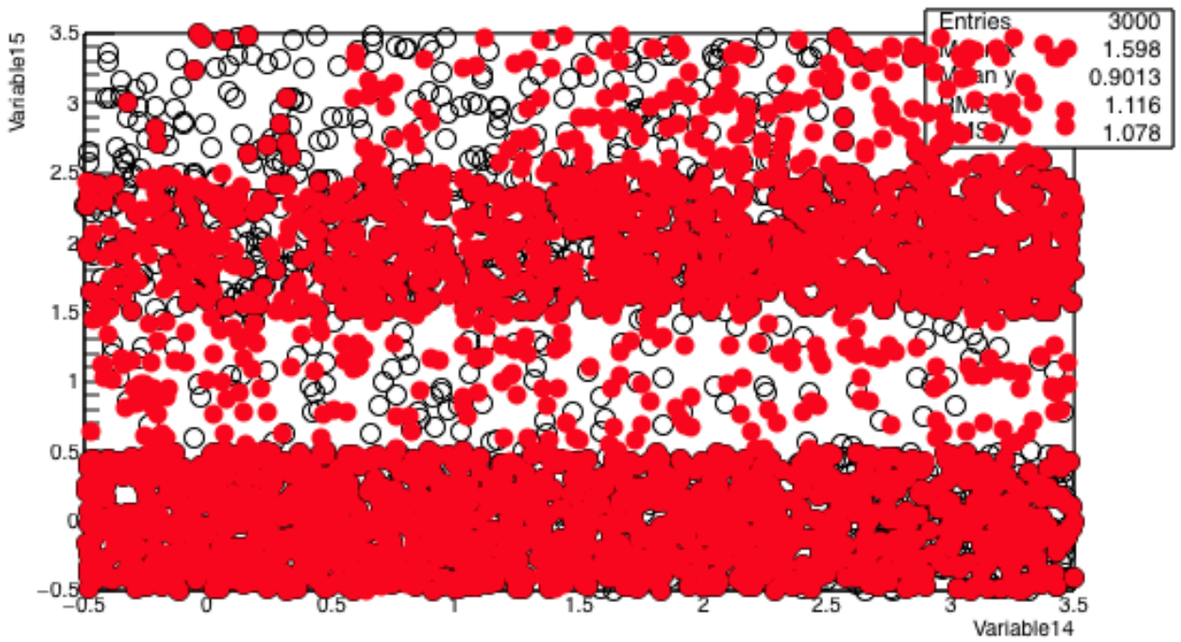


Exercise #2 Correlation Plots



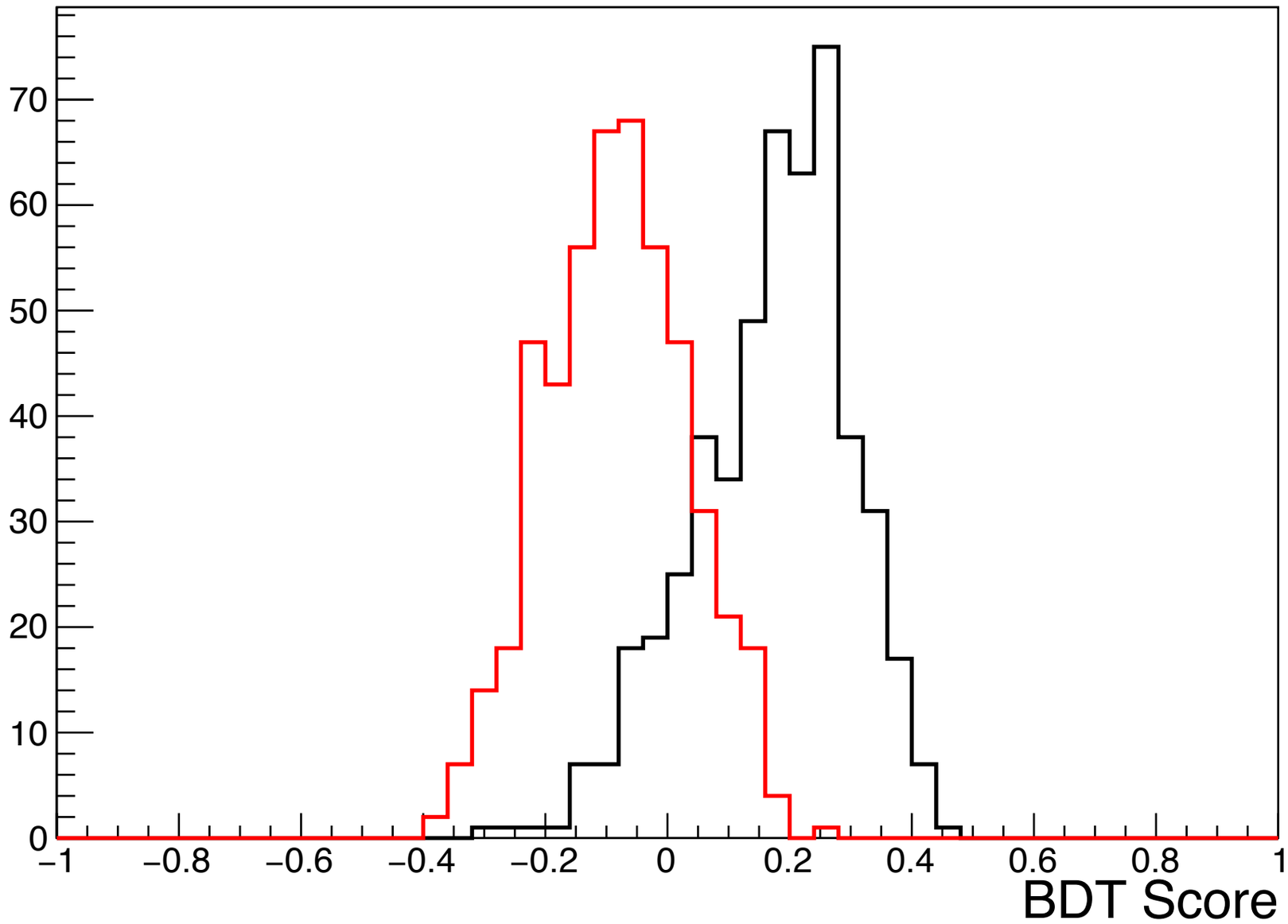






Exercise #2 BDT Scores

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



Next

- The following is one of the problems from the 2016 take-home exam
- For 2b the goal is to predict the cancer rate from an MVA trained on a blind data sample
  - 87%+ true positive rate got 100% credit
  - 85-87% true positive rate got 80% credit
  - 80-85% true positive rate got 60% credit
  - <80% true positive rate got 40% credit (for a valid effort)
- I have posted the 'solution' data set so you can test your own training acumen

Problem 2

*From 2016 Exam

- A cancer study in 1991 conducted in Wisconsin collected data from ~700 patients. There were 9 variables associated with the digitized image of a fine needle aspirate biopsy sample of a tissue mass. Each variable has discrete values from 1-10. There was also the patient identifier (code number) and whether the sample mass was ultimately benign or malignant.

#	Attribute	Domain/Range
1.	Sample code number	id number
2.	Clump Thickness	1 - 10
3.	Uniformity of Cell Size	1 - 10
4.	Uniformity of Cell Shape	1 - 10
5.	Marginal Adhesion	1 - 10
6.	Single Epithelial Cell Size	1 - 10
7.	Bare Nuclei	1 - 10
8.	Bland Chromatin	1 - 10
9.	Normal Nucleoli	1 - 10
10.	Mitoses	1 - 10
11.	Class:	(2 for benign, 4 for malignant)

Problem 2a

*From 2016 Exam

- There are two files online: training data and blind data
  - The following training data includes the aforementioned variables as well as whether the biopsy was benign or malignant (www.nbi.dk/~koskinen/Teaching/AdvancedMethodsInAppliedStatistics2016/data/breast-cancer-wisconsin_train-test.txt)
  - The following data includes the same variables, but the information of whether the biopsy was benign or malignant has been removed, i.e. a blind sample, (www.nbi.dk/~koskinen/Teaching/AdvancedMethodsInAppliedStatistics2016/data/breast-cancer-wisconsin_mod_real.txt)
- Using some method (straight cuts, support vector machine, boosted decision tree, etc.) and the training data, come up with a classification algorithm which uses the 9 variables to identify malignant and benign tissue samples
  - Note: the separate variables can be assumed to have the same features and shape between the training and blind sample

Problem 2a cont.

*From 2016 Exam

- With a developed algorithm, run the classifier over the training sample and calculate the efficiency (true positive rate) of identifying a malignant mass
  - It is possible to get 100% true positive rate, provided the method is overtrained
    - But, you will have to use the same settings for classifying the blind sample in Problem 2b
- Calculate the overall classification true positive rate for the whole training sample
  - (classified_true_malignant + classified_true_benign)/ (total_trainingtest_sample)

Problem 2b

*From 2016 Exam

- Using the same setting(s) as developed in Problem 2a, run the classifier over all the entries in the blind sample (breast-cancer-wisconsin_mod_real.txt)
  - Produce a text file which contains only the ID of the samples which your classifier classifies as malignant (last_name.malignant_ID.txt)
  - Produce a text file which contains only the ID of the samples which your classifier classifies as benign (last_name.benign_ID.txt)
  - Basic text files. No Microsoft Word documents, Adobe PDF, or any other extraneous text editor formats. Only a single ID number per line in the text file that can be easily read by numpy.loadtxt().
    - Example online at http://www.nbi.dk/~koskinen/Teaching/AdvancedMethodsInAppliedStatistics2016/data/koskinen.benign_ID.txt
  - Any and all duplicates, i.e. two samples with the same ID, should be kept and included in the text files and analysis

Additional Information

- Gradient Boosting
  - Original paper covering gradient boosting https://projecteuclid.org/euclid.aos/1013203451
  - XGBoost - “Extreme Gradient Boost”
    - Paper from XGBoost authors at https://doi.org/10.1145/2939672.2939785
    - Weight quantile sketch algorithm in XGBoost at https://homes.cs.washington.edu/~tqchen/pdf/xgboost-supp.pdf
  - Nice webpage covering explanation of gradient boosting concept https://explained.ai/gradient-boosting/
- Adaptive Boosting
  - AdaBoost is one of the most common boosting algorithms https://doi.org/10.1006/jcss.1997.1504