Introduction to Multivariate Methods

Classification and Function Approximation

Harrison B. Prosper

Florida State University

Bari Lectures

30, **31** May, 1 June 2016

Outline

- Lecture 1
  - Introduction
  - Classification
  - Grid Searches
  - Decision Trees
- Lecture 2
  - Boosted Decision Trees
- Lecture 3
  - Neural Networks
  - Bayesian Neural Networks

Recap: Goal

The goal of a typical multivariate method is to approximate the mapping of “inputs”, or “features”,

$$x = (x_1, x_2, \ldots, x_n)$$

to “outputs”, or “responses”, y, where

$$y = f(x)$$

assuming some specific class $\{f\}$ of functions, together with some constraints on this class, e.g., the functions should be smooth.

Example: Wine Tasting

Today, we shall explore a method called boosted decision trees using the wine tasting example based on data by Cortez et al.*

http://www.vinhoverde.pt/en/history-of-vinho-verde

* P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis.
Modeling wine preferences by data mining from physicochemical properties.
In Decision Support Systems, Elsevier, 47(4):547-553. ISSN: 0167-9236.

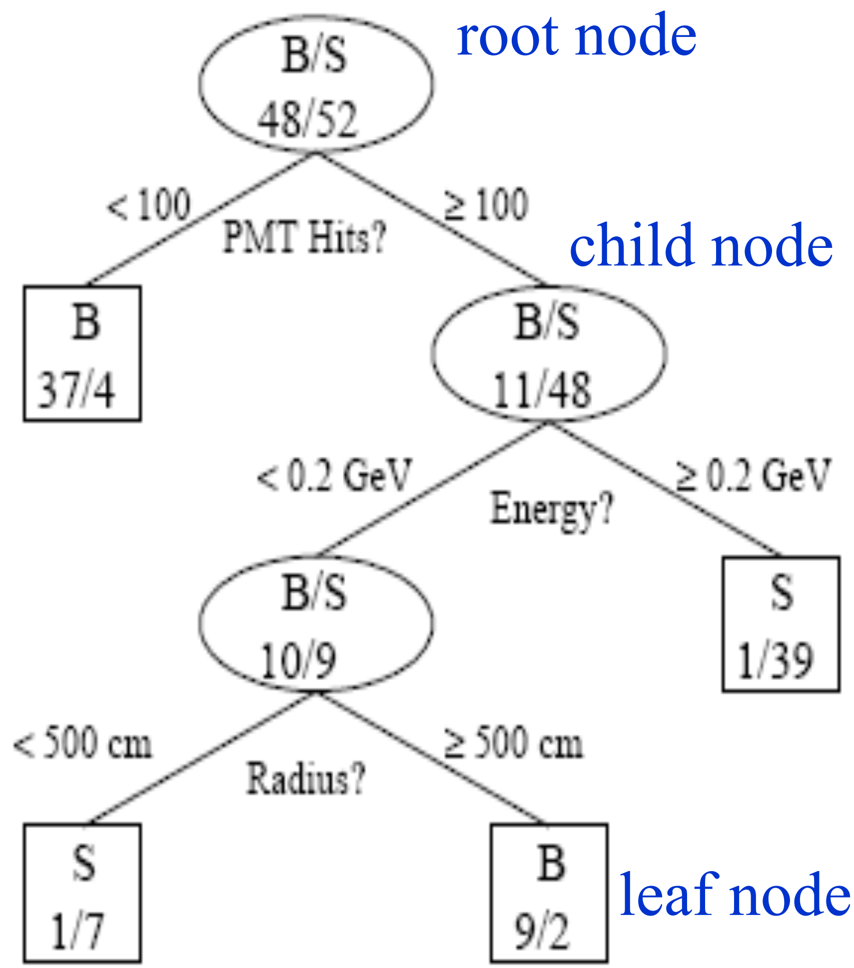
Recap: Decision Trees

Recap: Decision Trees

Decision tree:
a sequence of if then else statements.

Basic idea: recursively partition the space $\{x\}$ into regions of increasing purity.

Geometrically, a decision tree is a *d-dimensional* histogram in which the bins are built using recursive *binary* partitioning.

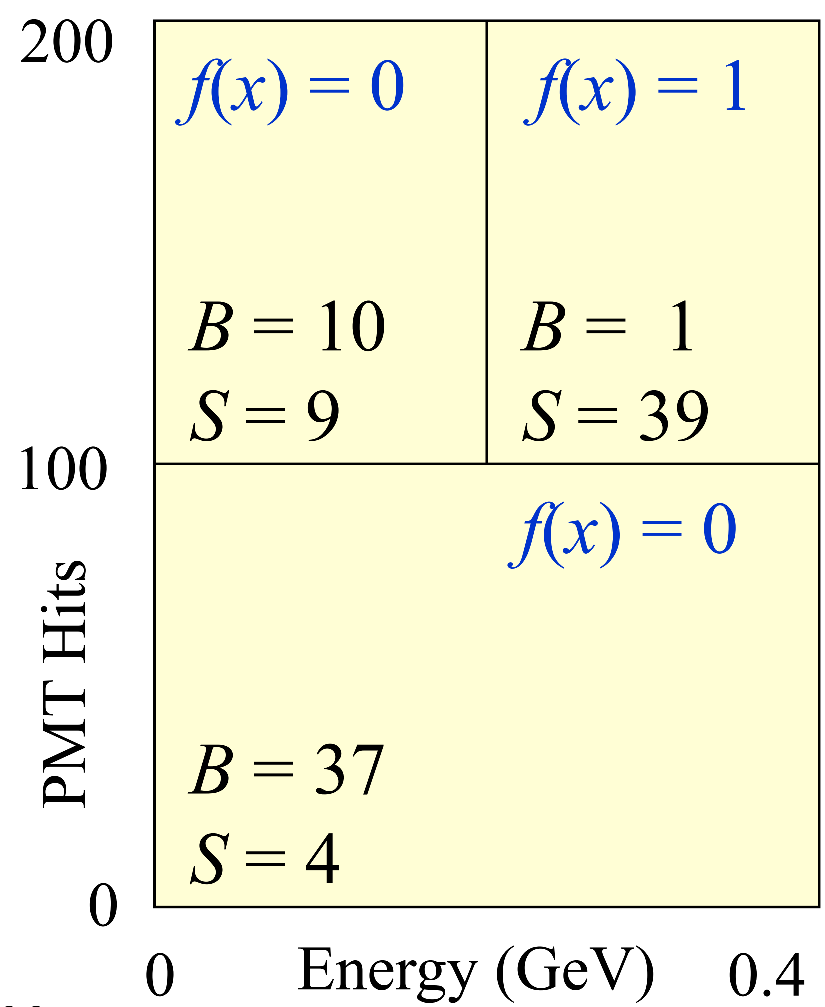


MiniBoone, Byron Roe

Recap: Decision Trees

To each bin, we associate the value of the function $f(x)$ to be approximated.

That way, we arrive at a piecewise constant approximation of $f(x)$.



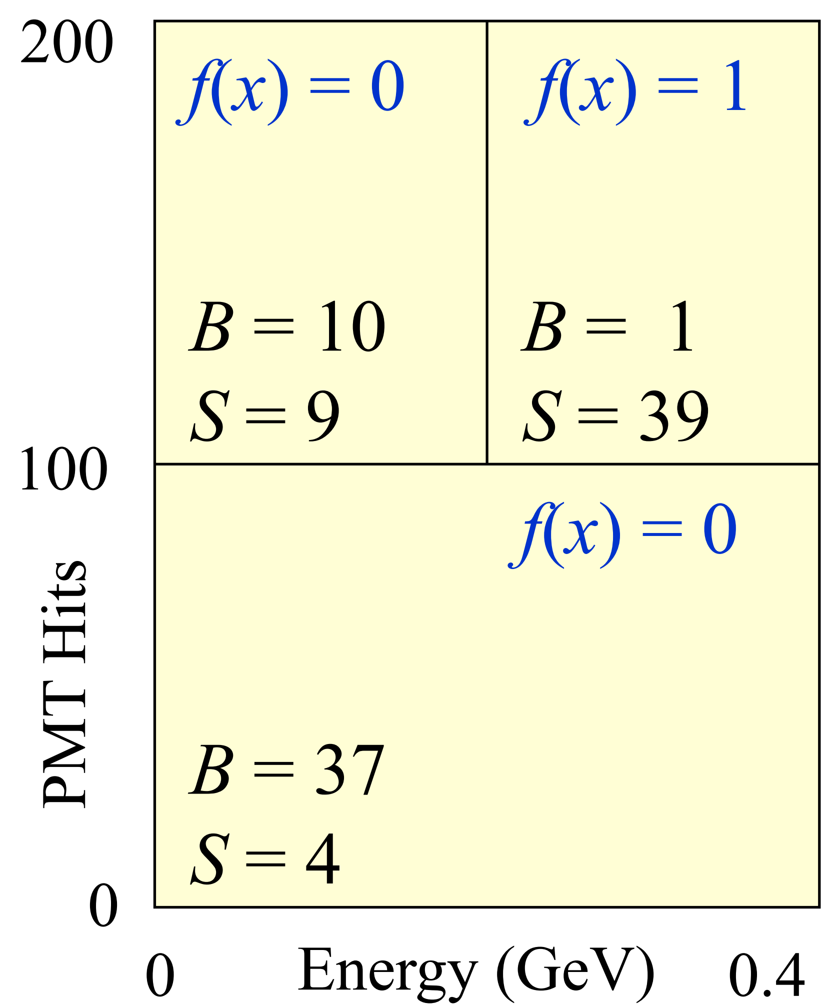
MiniBoone, Byron Roe

Decision Trees

For each variable, find the best partition ("cut"), defined as the one that yields the greatest *decrease* in impurity

= **Impurity** (parent bin)
– **Impurity** ("left"-bin)
– **Impurity** ("right"-bin)

Then choose the best partition among all partitions, and repeat with each child bin



Decision Trees

The most common impurity measure is the Gini index

(Corrado Gini, 1884-1965):

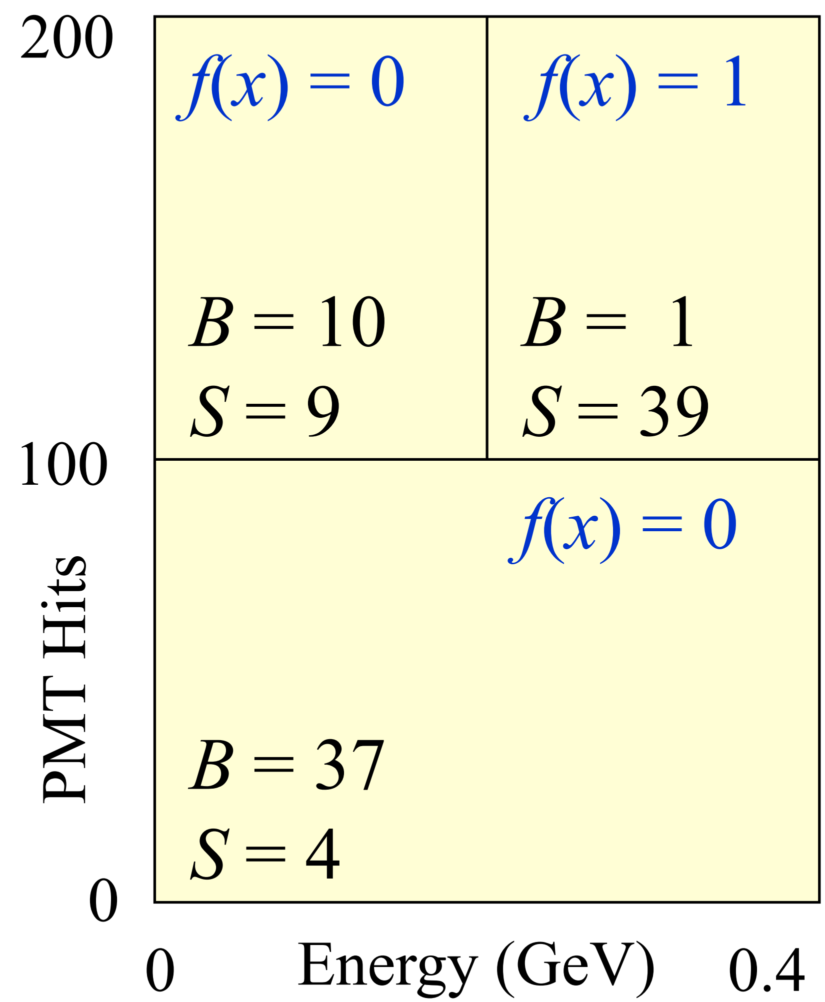
Gini index = $p\,(1 - p)$

where p is the purity

$p = S\,/\,(S + B)$

$p = 0$ or 1 = maximal purity

$p = 0.5$ = maximal impurity



Boosted Decision Trees

Introduction

Until relatively recently, the goal of researchers who worked on classification methods was to construct directly a *single* high performance classifier.

However, in 1997, AT&T researchers Y. Freund and R.E. Schapire [Journal of Computer and Sys. Sci. **55** (1), 119 (1997)], showed that it was possible to build highly effective classifiers by combining many weak ones!

JOURNAL OF COMPUTER AND SYSTEM SCIENCES **55**, 119–139 (1997)
ARTICLE NO. SS971504

This was the first successful method to *boost* (i.e., enhance) the performance of poorly performing classifiers by averaging them.

A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting*

Yoav Freund and Robert E. Schapire†
AT&T Labs, 180 Park Avenue, Florham Park, New Jersey 07932

Received December 19, 1996

Averaging Weak Learners

Suppose you have a collection of classifiers $f(x, w_k)$, which, individually, perform only marginally better than random guessing. Such classifiers are called weak learners.

It is possible to build highly effective classifiers by *averaging* many weak learners:

$$f(x) = a_0 + \sum_{k=1}^{K} a_k f(x, w_k)$$

Jeromme Friedman & Bogdan Popescu (2008)

Averaging Weak Learners

The most popular methods (used mostly with decision trees) are:

- Bagging: each tree is trained on a bootstrap* sample drawn from the training set
- Random Forest: bagging with randomized trees
- Boosting: each tree trained on a different reweighting of the training set

*A bootstrap sample is a sample of size N drawn, *with replacement*, from another of the same size. Duplicates can occur and are allowed.

Adaptive Boosting

The AdaBoost algorithm of Freund and Schapire uses decision trees $f(x, \mathbf{w})$ as the weak learners, where $\mathbf{w}$ are weights assigned to the objects to be classified, each associated with a label $y = \pm 1$, e.g., $+1$ for good wine, -1 for bad.

The value assigned to each leaf of $f(x, \mathbf{w})$ is also ± 1.

Consequently, for object n, associated with values (y_n, x_n), the product

$f(x_n, \mathbf{w})\, y_n > 0$ for a correct classification

$f(x_n, \mathbf{w})\, y_n < 0$ for an incorrect classification

Next, we consider the actual boosting algorithm…

Y. Freund and R.E. Schapire. Journal of Computer and Sys. Sci. **55** (1), 119 (1997)

Adaptive Boosting

Initialize weights $\boldsymbol{w}$ in training set (e.g., setting each to 1/N)

For $k = 1$ to K:

1. Create a decision tree $f(x, \boldsymbol{w})$ using the current weights.
2. Compute its error rate ε on the *weighted* training set.
3. Compute $\alpha = \ln(1 - \varepsilon) / \varepsilon$ and store as $\alpha_k = \alpha$
4. Update each weight w_n in the training set as follows: new-$w_n = w_n \exp[-\alpha_k f(x_n, \boldsymbol{w})\, y_n]/A$, where A is a normalization constant such that $\sum$new-$w_n = 1$. Since $f(x_n, \boldsymbol{w})\, y_n < 0$ for an incorrect classification, the weight of misclassified objects is *increased*.

At the end, compute the average $f(x) = \sum \alpha_k\, f(x, \boldsymbol{w}_k)$

Y. Freund and R.E. Schapire. Journal of Computer and Sys. Sci. **55** (1), 119 (1997)

Adaptive Boosting

AdaBoost is a very non-intuitive algorithm. However, soon after its invention Friedman, Hastie and Tibshirani showed that the algorithm is mathematically equivalent to minimizing the following risk (or cost) function

$$R(F) = \int p(x,y) \exp(-yF(x))\, dx\, dy,$$

$$\text{where } F(x) = \sum_{k=1}^{K} \alpha_k f(x, w_k)$$

which implies that $$D(x) = \frac{1}{1 + \exp(-2F(x))}$$

can be interpreted as a probability, even though F cannot!

J. Friedman, T. Hastie and R. Tibshirani, ("Additive logistic regression: a statistical view of boosting," The Annals of Statistics, 28(2), 377-386, (2000))

Example: Wine Tasting

Let's use the AdaBoost algorithm to build a classifier that can distinguish between good wines and "bad" wines from the Vinho Verde area of Portugal using the data from Cortez *et al.*

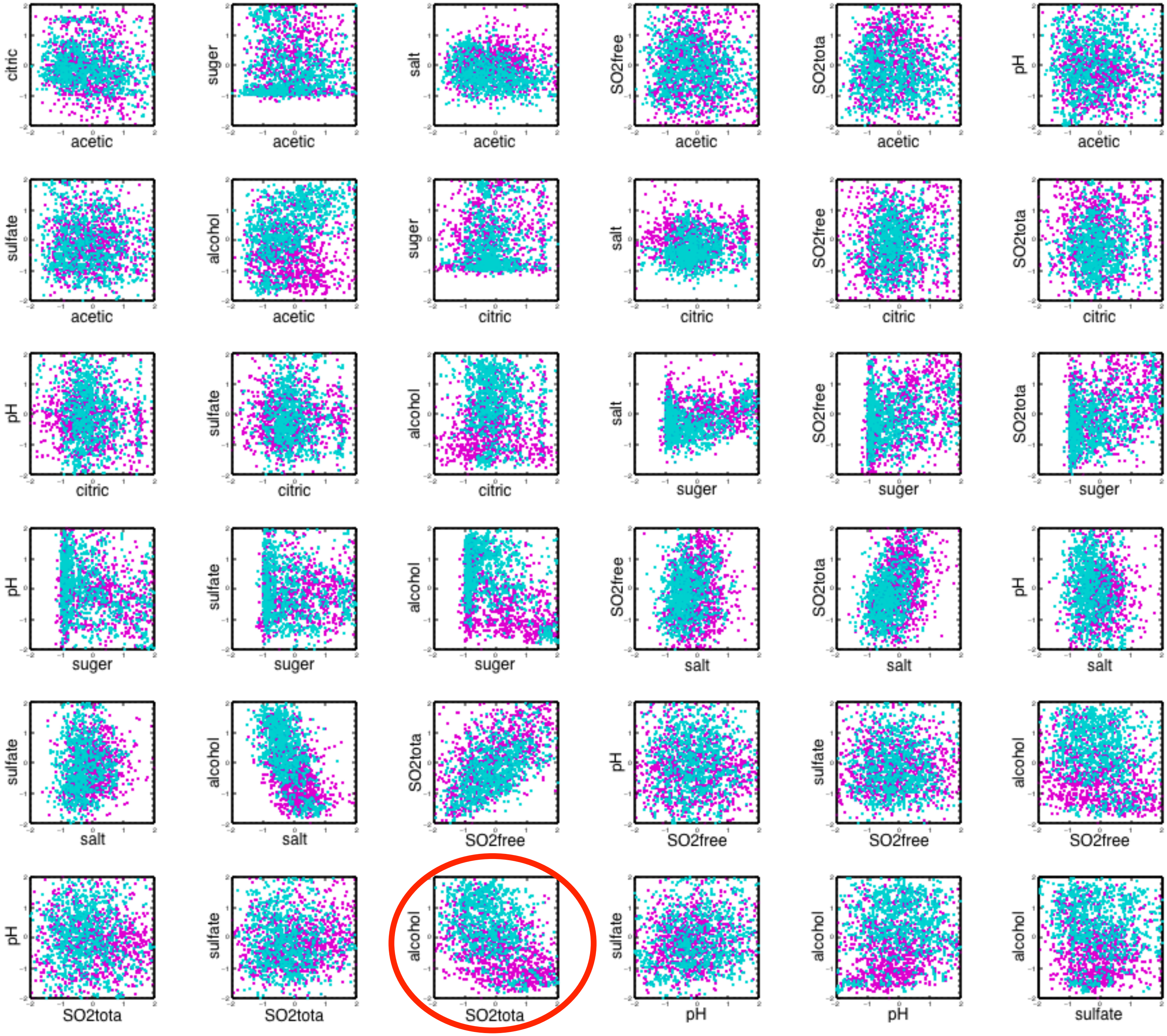
We'll define a good wine as one with rating ≥ 0.7 on a scale from 0 to 1, where 1 is a wine from Heaven and 0 is a wine from Hell!

First, let's look at the training data…

Wine Tasting

Data: [Cortez *et al.*, 2009].

variables	description
acetic	acetic acid
citric	citric acid
sugar	residual sugar
salt	NaCl
SO2free	free sulfur dioxide
SO2tota	total sulfur dioxide
pH	pH
sulfate	potassium sulfate
alcohol	alcohol content
quality	(between 0 and 1)

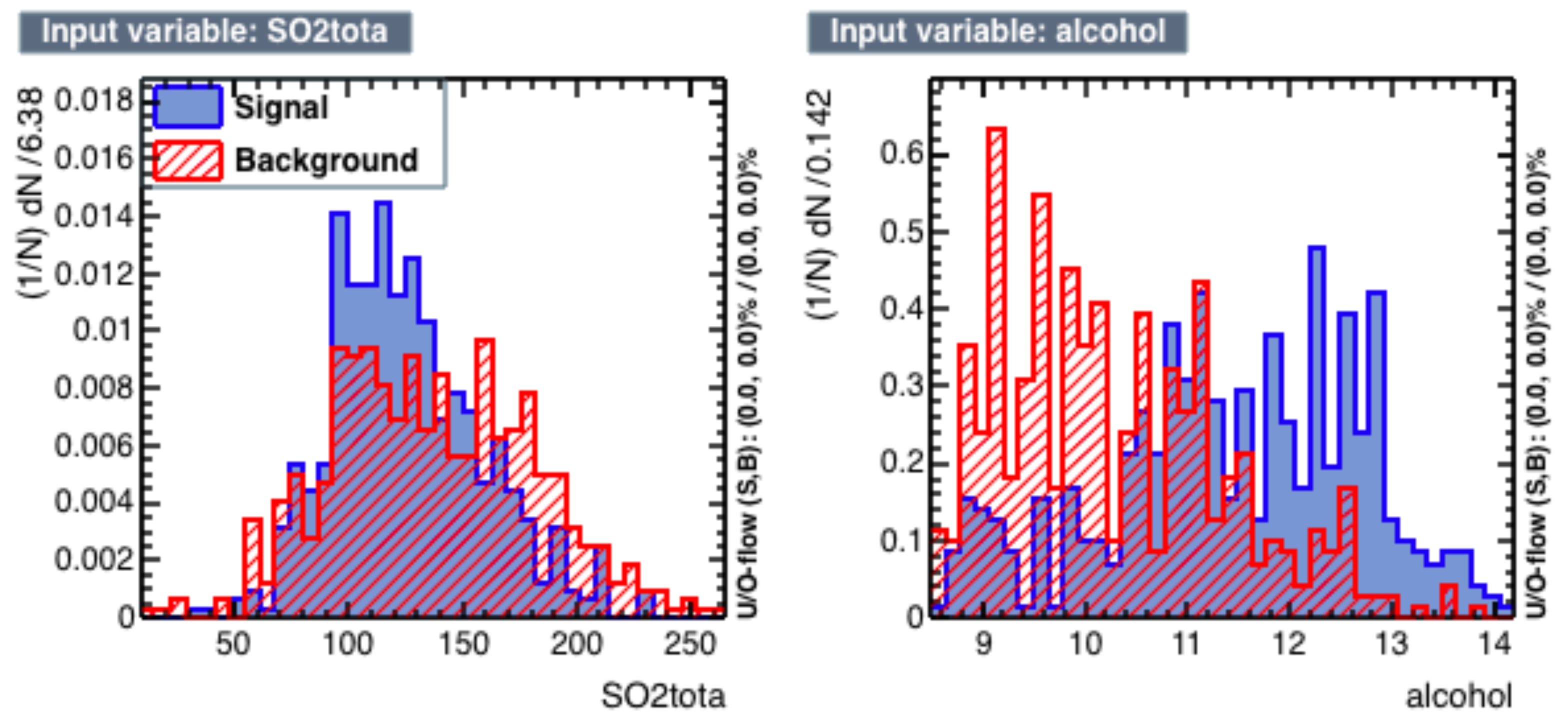

citric
acetic
suger
acetic
salt
acetic
SO2free
acetic
SO2tota
acetic
pH
acetic
sulfate
acetic
alcohol
acetic
suger
citric
salt
citric
SO2free
citric
SO2tota
citric
pH
citric
sulfate
citric
alcohol
citric
salt
suger
SO2free
suger
SO2tota
suger
pH
suger
sulfate
suger
alcohol
suger
SO2free
salt
SO2tota
salt
pH
salt
sulfate
salt
alcohol
salt
SO2tota
SO2free
pH
SO2free
sulfate
SO2free
alcohol
SO2free
pH
SO2tota
sulfate
SO2tota
alcohol
SO2tota
sulfate
pH
alcohol
pH
alcohol
sulfate

Example: Wine Tasting

To make visualization easier, we'll use only two variables:

SO2tota: the total sulfur dioxide content (mg/dm^3)

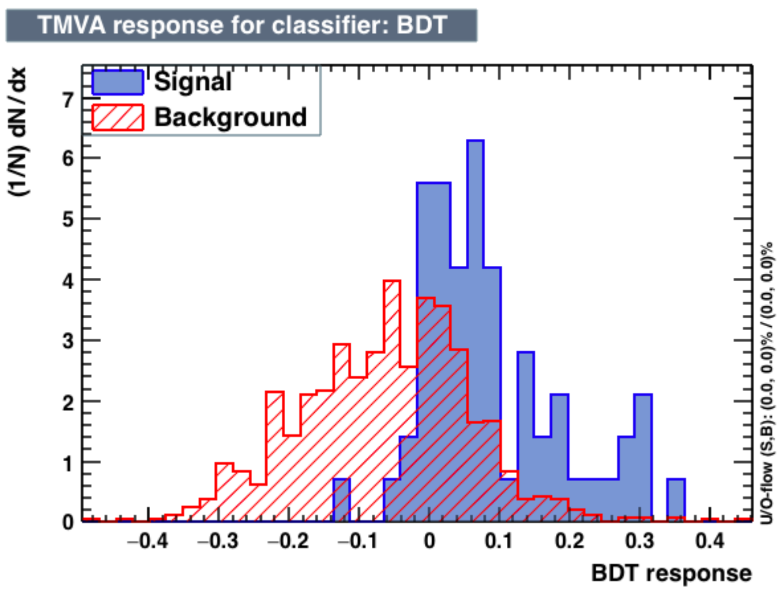
alcohol: alcohol content (% volume)



Results

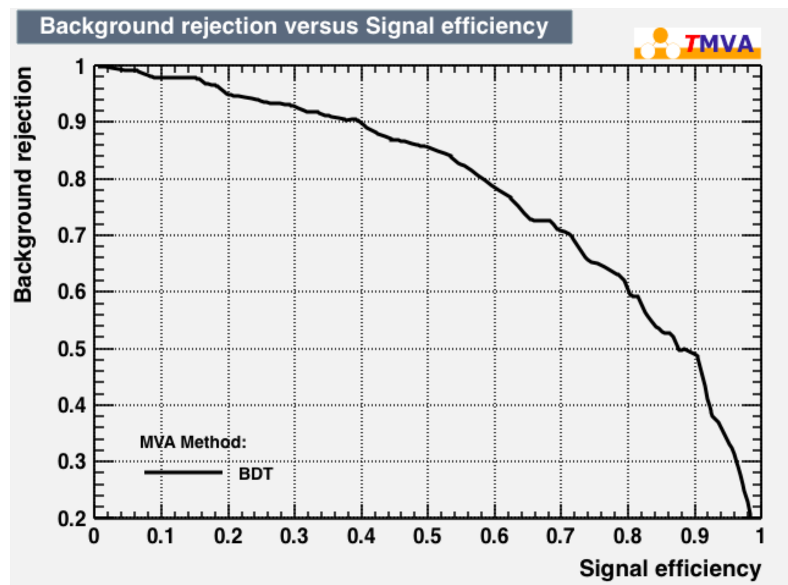
x = SO2tota

y = alcohol



Distribution of BDT response

$$BDT(x, y) = \sum_{k=0}^{99} a_k f(x, y, w_k)$$



Fraction of bad wine rejected for a given fraction of good wine accepted.

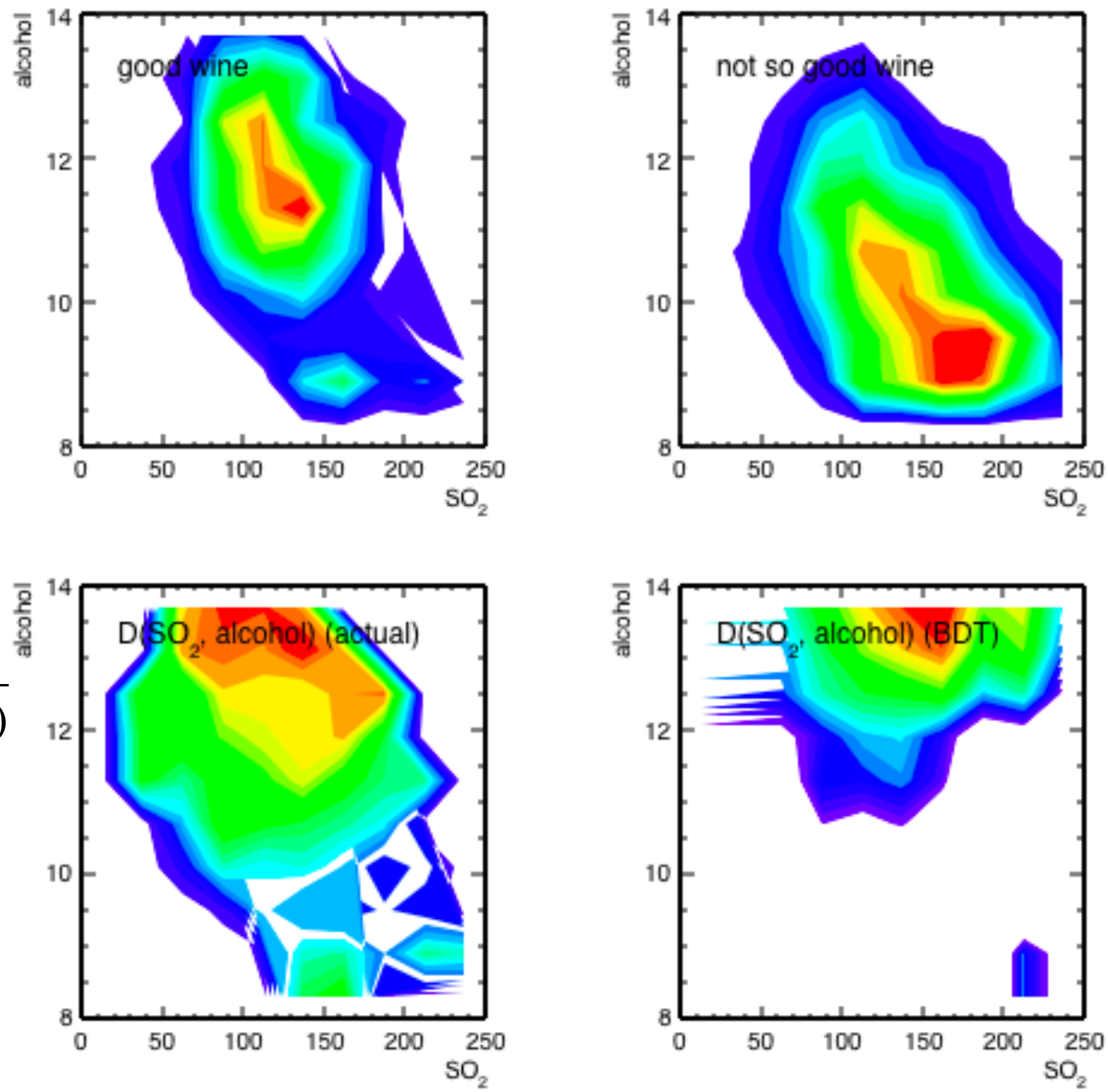
Results

The upper figures are density plots of the training data.

The lower plots are approximations of

$$D = \frac{p(x,y\,|\,good)}{p(x,y\,|\,good) + p(x,y\,|\,bad)}$$

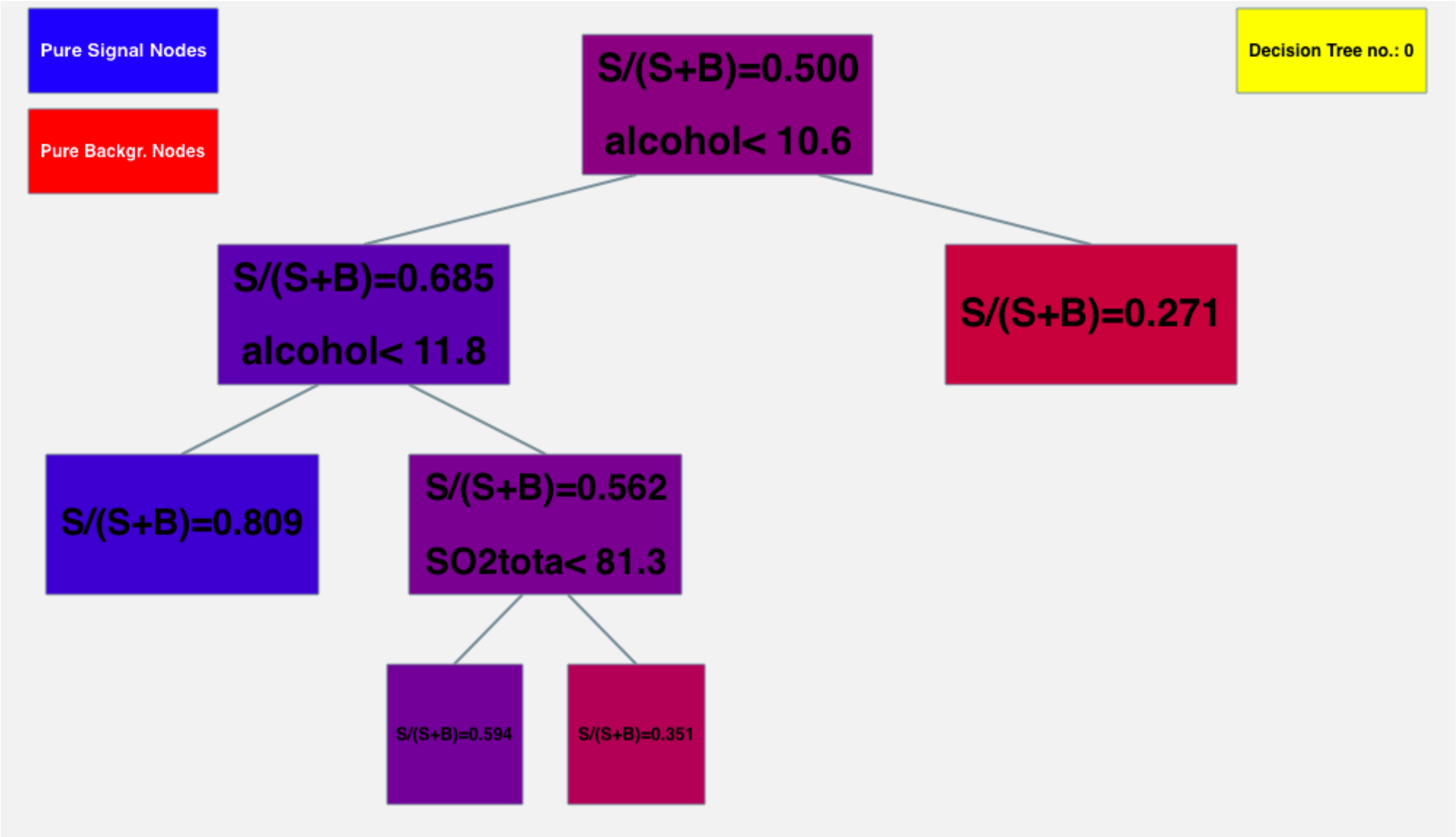
The left, uses 2-D histograms, the right uses the BDT.

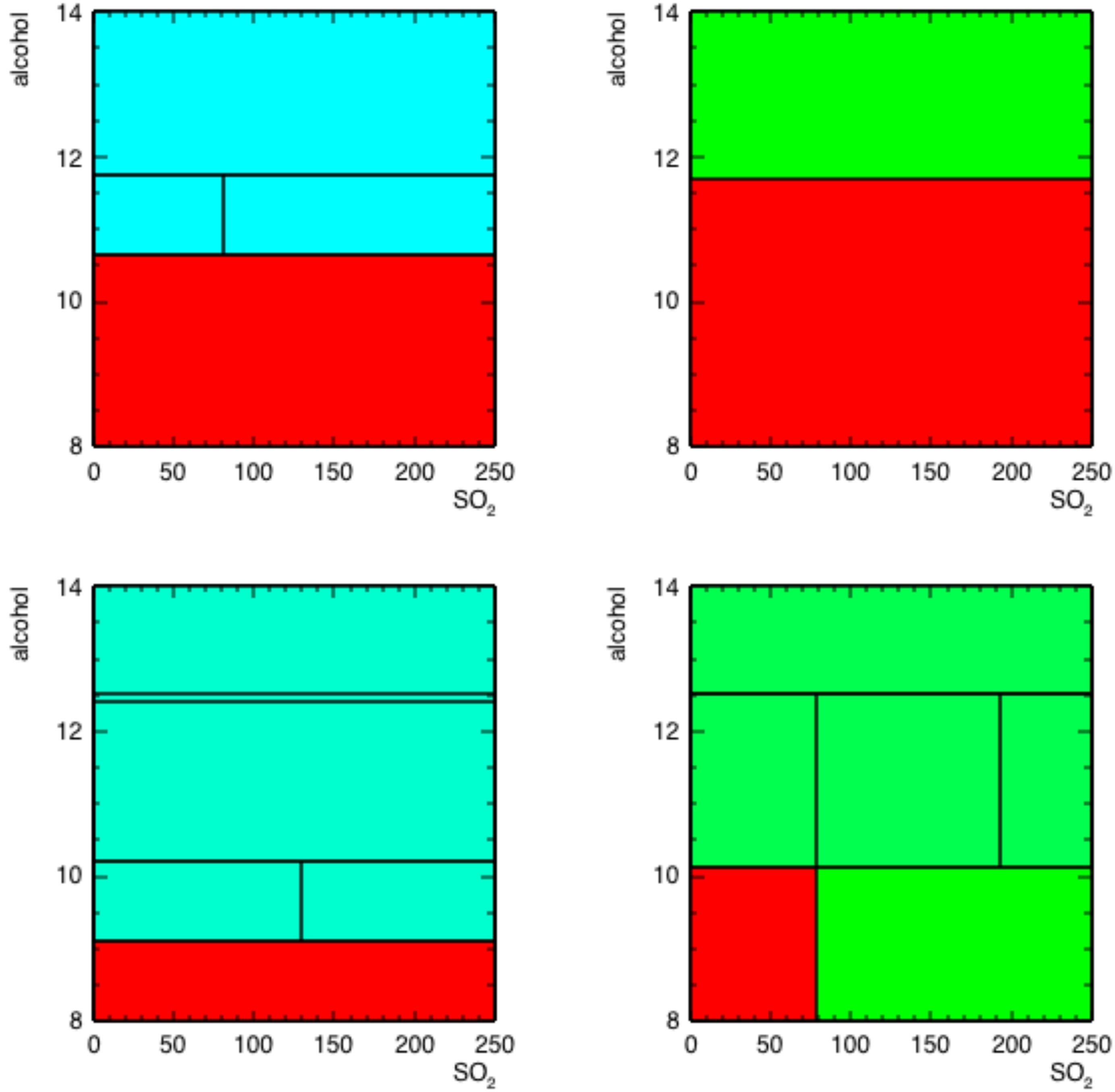


Results

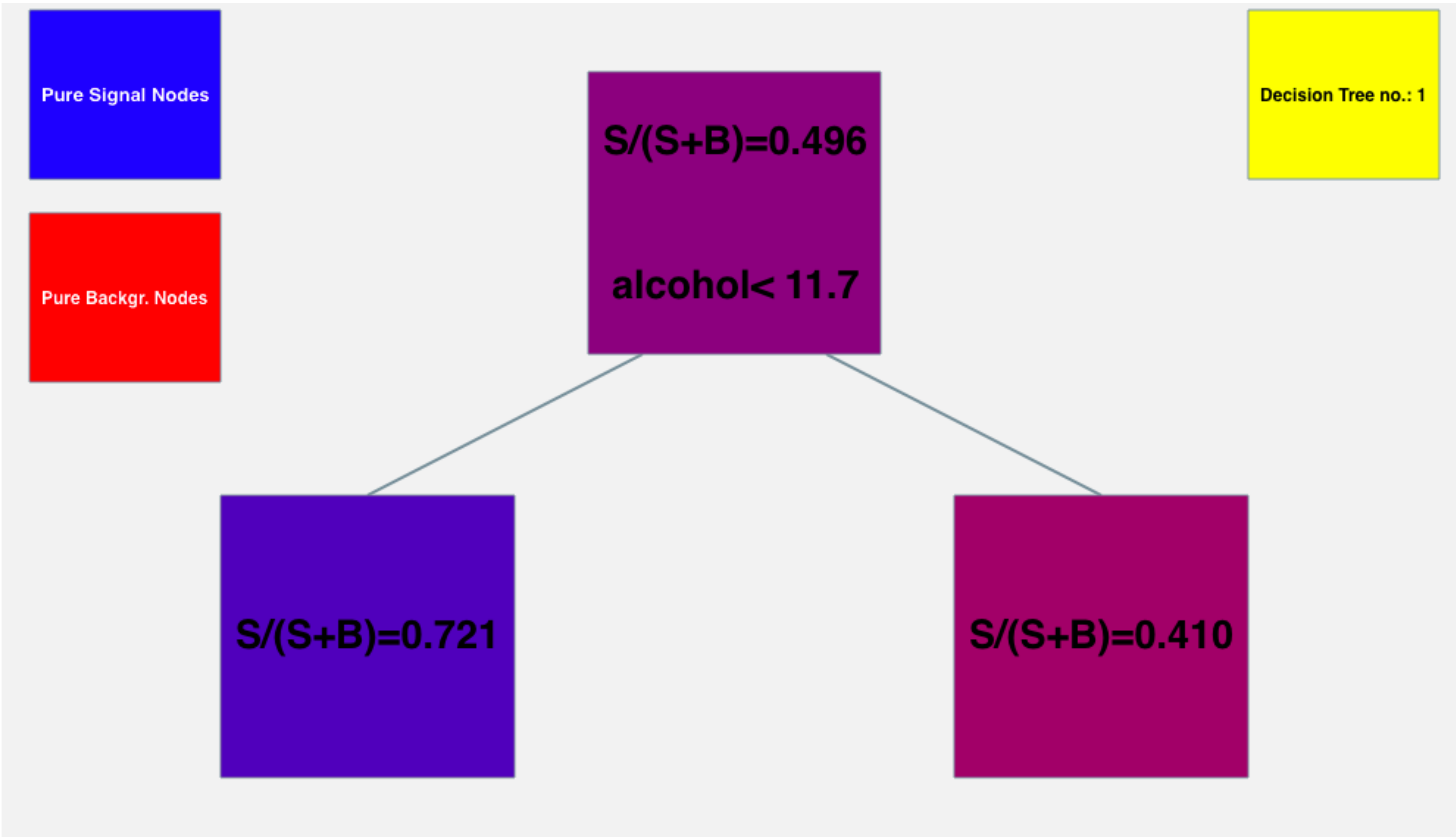
Let's dig more deeply…

Tree 0

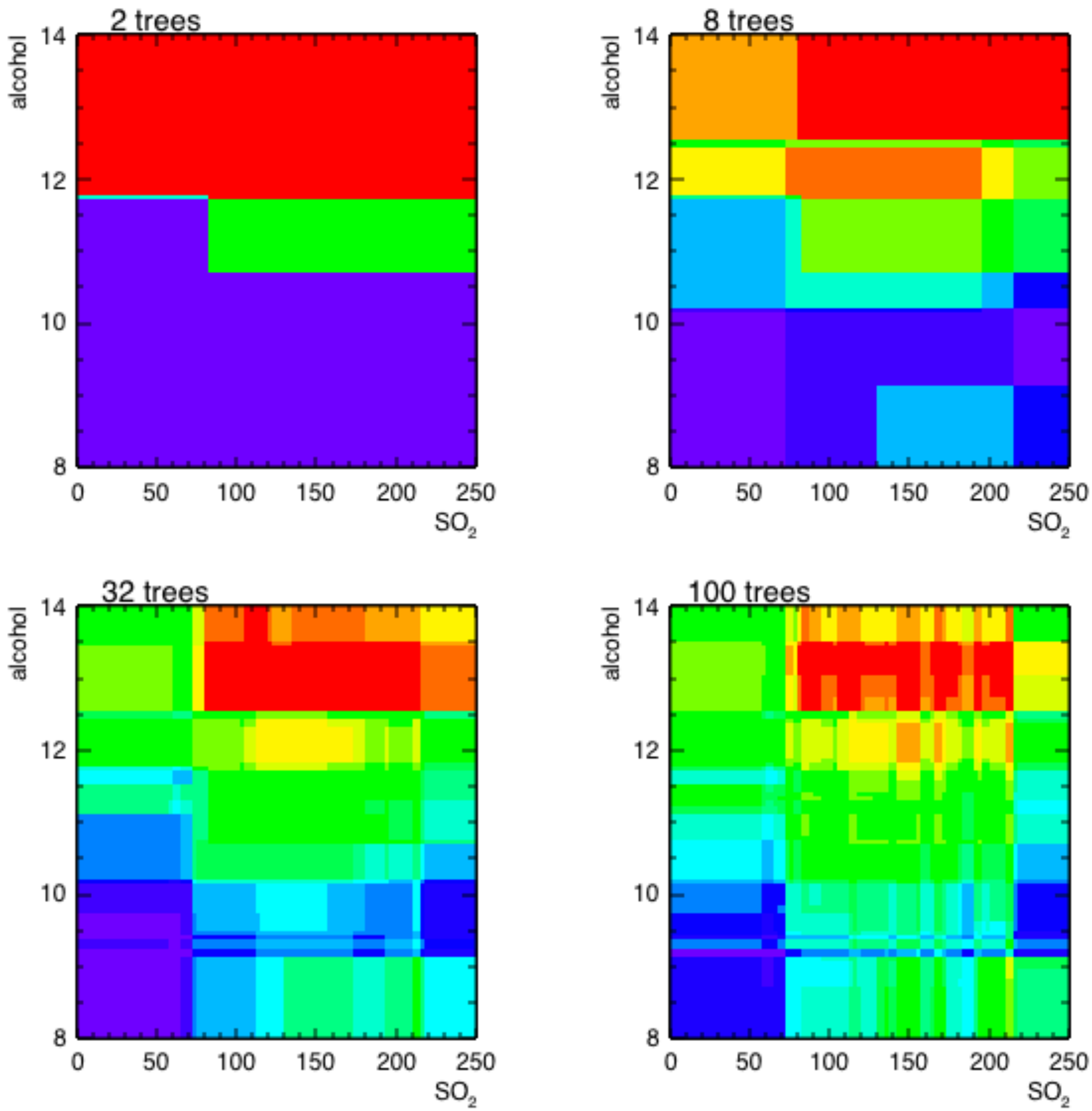

Pure Signal Nodes
Pure Backgr. Nodes
Decision Tree no.: 0
S/(S+B)=0.500
alcohol< 10.6
S/(S+B)=0.685
alcohol< 11.8
S/(S+B)=0.271
S/(S+B)=0.809
S/(S+B)=0.562
SO2tota< 81.3
S/(S+B)=0.594
S/(S+B)=0.351


alcohol
SO2
14
12
10
8
0
50
100
150
200
250

Tree 1



Pure Signal Nodes
Pure Backgr. Nodes
Decision Tree no.: 1
S/(S+B)=0.496
alcohol< 11.7
S/(S+B)=0.721
S/(S+B)=0.410


2 trees
8 trees
32 trees
100 trees
alcohol
SO2
14
12
10
8
0
50
100
150
200
250

Summary

- It is possible to average many relatively crude decision trees to obtain a better approximation to the function

$$D(x,y) = \frac{p(x,y \mid good)}{p(x,y \mid good) + p(x,y \mid bad)}$$

- Tomorrow, we shall consider examples of classification and regression using neural networks.