Chapter 11

Classification Algorithms and Regression Trees

The next four paragraphs are from the book by Breiman et. al.

At the university of California, San Diego Medical Center, when a heart attack patient is admitted, 19 variables are measured during the first 24 hours. They include BP, age and 17 other binary covariates summarizing the medical symptoms considered as important indicators of the patient's condition.

The goal of a medical study can be to develop a method to identify high risk patients on the basis of the initial 24-hour data.

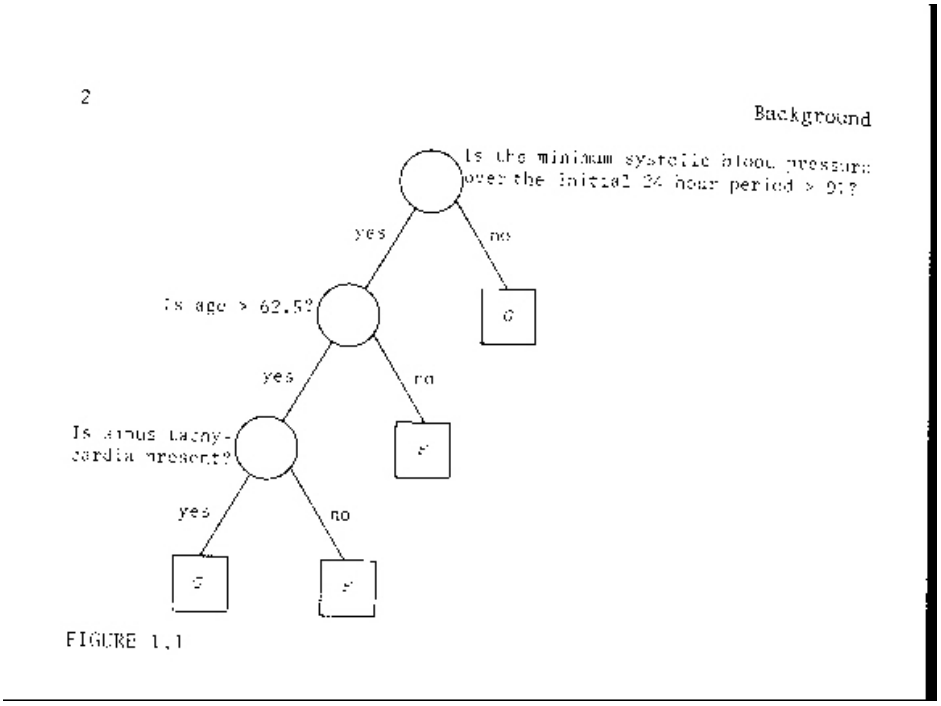


Figure 11.1: Ad-hoc decision tree for risk.

The next figure shows a picture of a tree structured classification rule that was produced in the study. The letter F means no high and the letter G means high risk.

How can we use data to construct trees that give us useful answers. There is a large amount of work done in this type of problem. We will give an introductory description in this section.

The material here is based on lectures by Ingo Ruczinski.

11.1 Classifiers as Partitions

Notice that in the example above we predict a positive outcome if both blood pressure is high **and** age is higher than 62.5. This type of interaction is hard to describe by a regression model. If you go back and look at the methods we presented you will notice that we rarely include interaction terms mainly because of the curse of dimensionality. There are too many interactions to consider and too many ways to quantify their effect. Regression trees thrive on such interactions. What is a curse for parametric and smoothing approaches is a blessing for regression trees.

A good example is the following olive data:

- 572 olive oils were analyzed for their content of eight fatty acids (palmitic, palmitoleic, stearic, oleic, linoleic, arachidic, linolenic, and eicosenoic).
- There were 9 collection areas, 4 from Southern Italy (North and South Apulia, Calabria, Sicily), two from Sardinia (Inland and Coastal) and 3 from Northern Italy (Umbria, East and West Liguria).
- The concentrations of different fatty acids vary from up to 85% for oleic acid to as low as 0.01% for eicosenoic acid.
- See Forina M, Armanino C, Lanteri S, and Tiscornia E (1983). *Classification of olive oils from their fatty acid composition*. In Martens H and

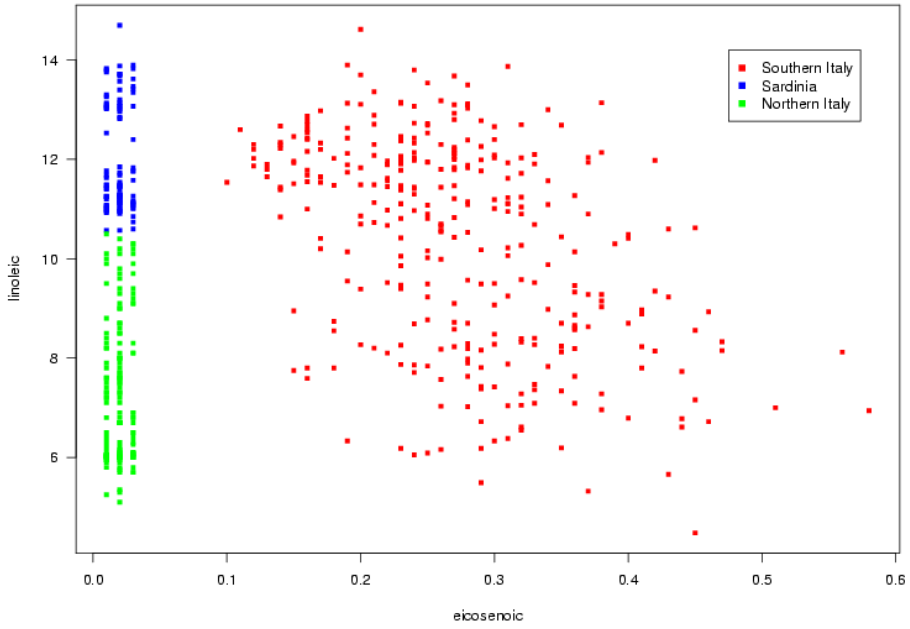


Figure 11.2: Olive data.

Russwurm Jr H, editors, Food Research and Data Analysis, pp 189-214. Applied Science Publishers, London.

The data look like this:

Notice that we can separate the covariate space so that we get perfect prediction without a very complicated “model”.

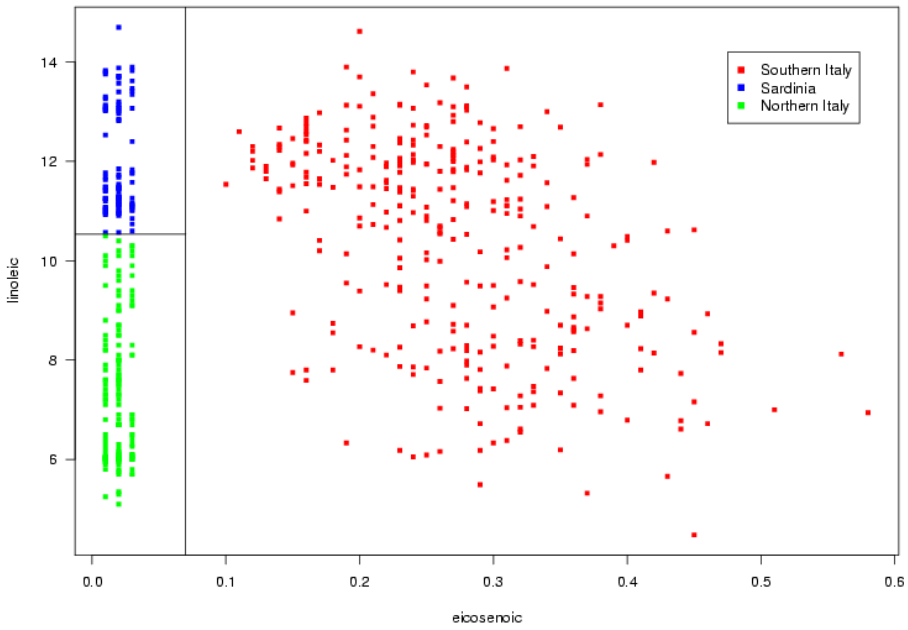


Figure 11.3: Simple partition with 0 training errors.

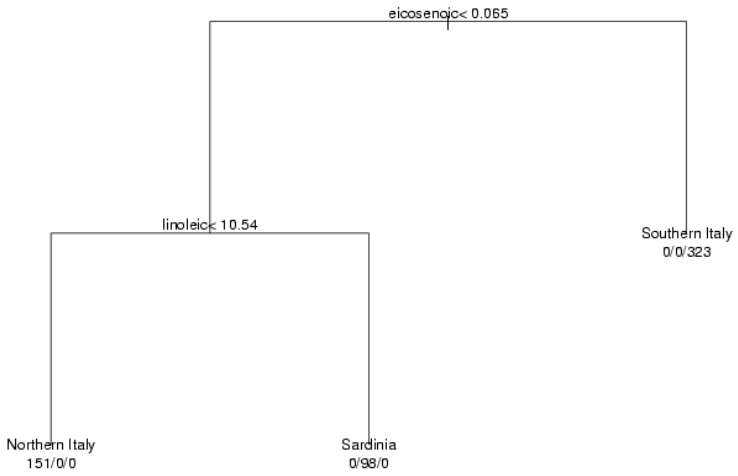


Figure 11.4: Tree that goes with previous plot.

The tree representation of this picture is

Partition such as this can also handle data where linear methods work well. A good (and very famous) example is Fisher's Iris Data:

However, none of the methods that we have describe permit a division of the space without using many parameters.

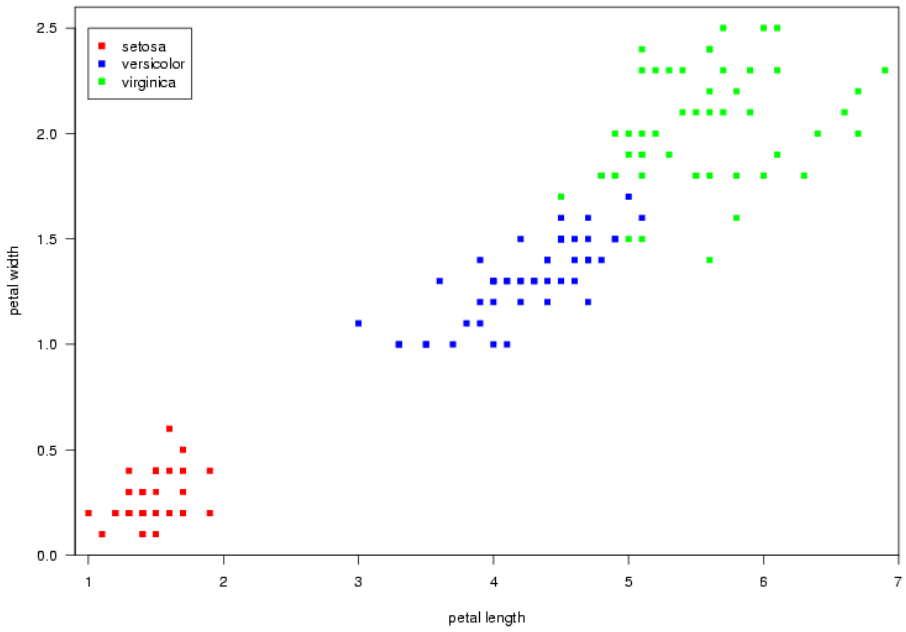


Figure 11.5: Iris Data.

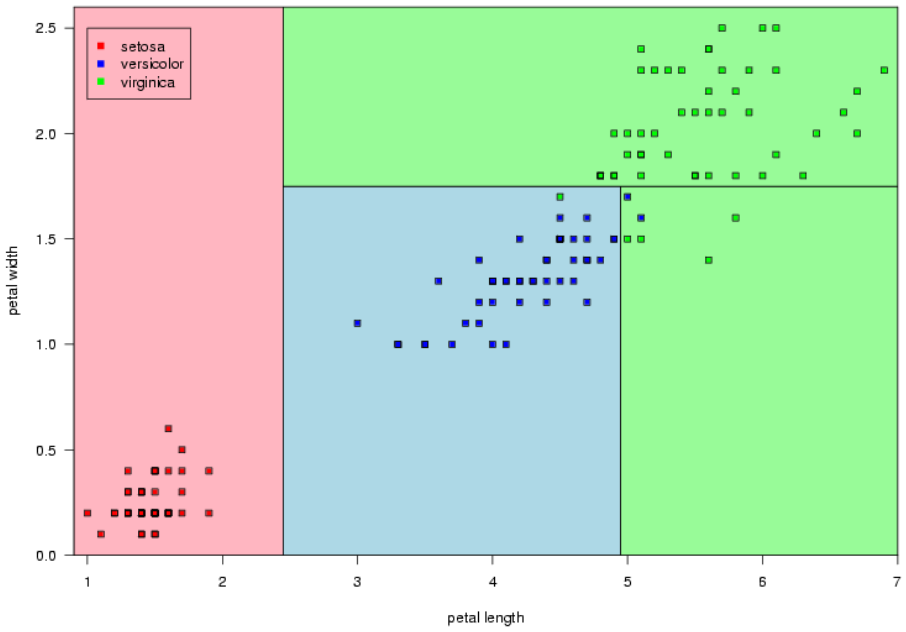


Figure 11.6: Simple partition with almost no training errors.

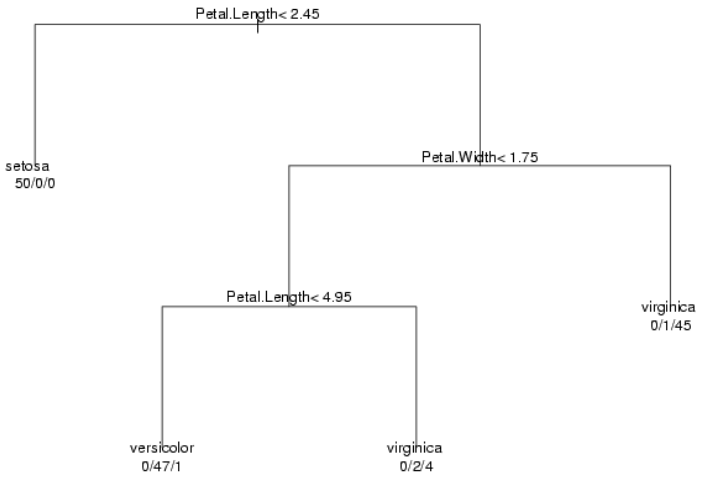


Figure 11.7: Tree for the previous plot.

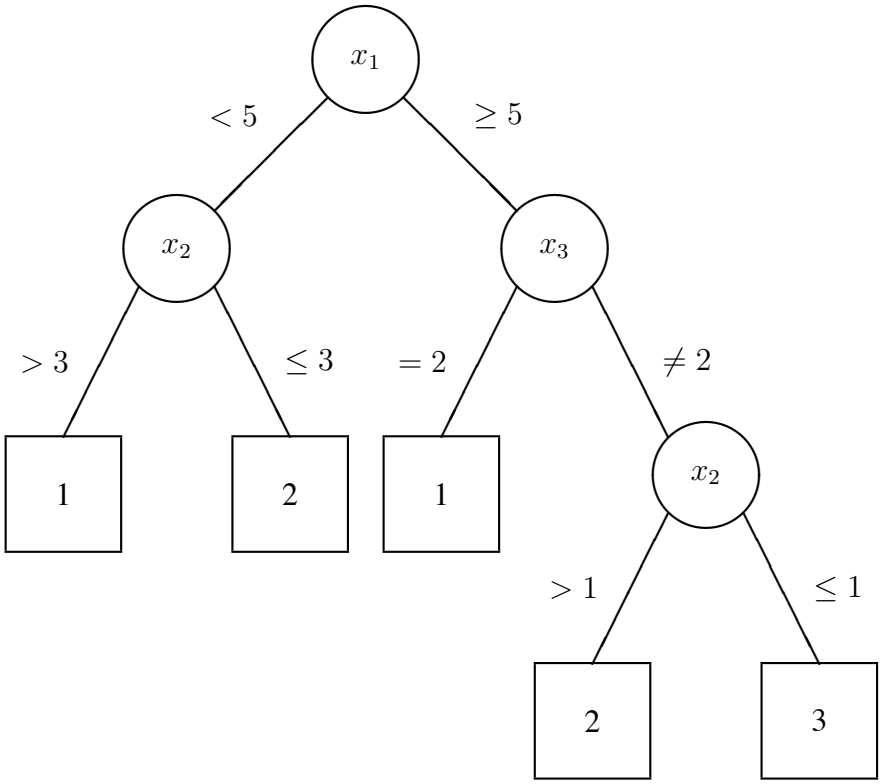
11.2 Trees

These data motivates the approach of partitioning the covariate space $\mathcal{X}$ into disjoint sets $A_1, \ldots, A_j$ with $\hat{G} = j$ for all $\mathbf{x} \in A_j$. There are too many ways of doing so we ways to make the approach more parsimonious. Notice that linear regression restricts divisions to certain planes.

Trees are a completely different way of partitioning. All we require is that the partition can be achieved by successive binary partitions based on the different predictors. Once we have a partition such as this we base our prediction on the average of the Ys in each partition. We can use this for both classification and regression.

11.2.1 Example of classification tree

Suppose that we have a scalar outcome, Y, and a p-vector of explanatory variables, X. Assume $Y \in \mathcal{K} = \{1, 2, \ldots, k\}$



The subsets created by the splits are called *nodes*. The subsets which are not split are called terminal nodes.

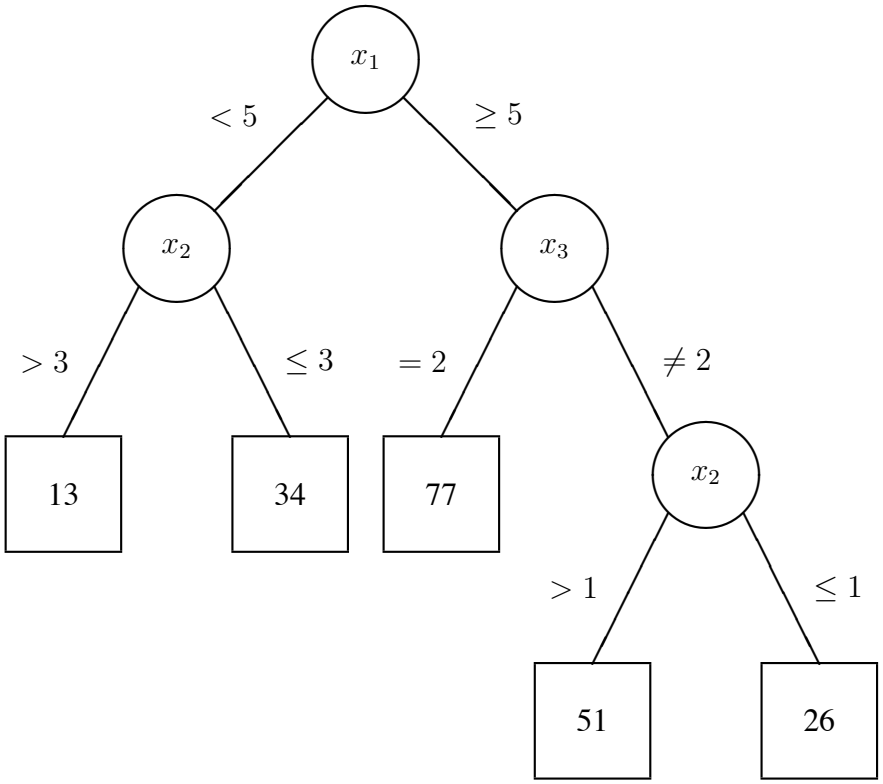
Each terminal nodes gets assigned to one of the classes. So if we had 3 classes we could get $A_1 = \mathcal{X}_5 \cup \mathcal{X}_9$, $A_2 = \mathcal{X}_6$ and $A_3 = \mathcal{X}_7 \cup \mathcal{X}_8$. If we are using the data we assign the class most frequently found in that subset of $\mathcal{X}$. We call these

classification tress.

A classification tree partitions the X-space and provides a predicted value, perhaps $\arg\max_s \Pr(Y = s|X \in A_k)$ in each region.

11.2.2 Example of regression tree

Again, suppose that we have a scalar outcome, Y, and a p-vector of explanatory variables, X. Now assume $Y \in \mathcal{R}$.



A regression tree partitions the X-space into disjoint regions A_k and provides a fitted value $\mathrm{E}(Y|X \in A_k)$ within each region.

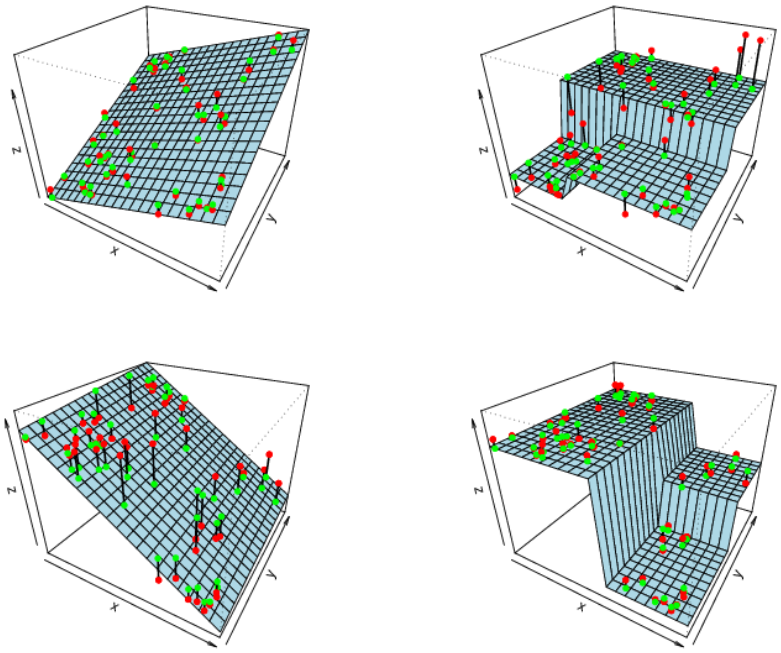


Figure 11.8: Comparison of CART and Linear Regression

11.2.3 CART versus Linear Models

See Figure 11.8.

11.3 Searching for good trees

In general the idea the following:

1. **Grow** an overly large tree using forward selection. At each step, find the *best* split. Grow until all terminal nodes either
   (a) have $< n$ (perhaps $n = 1$) data points,
   (b) are "pure" (all points in a node have [almost] the same outcome).
2. **Prune** the tree back, creating a nested sequence of trees, decreasing in complexity.

A problem in tree construction is how to use the training data to determine the binary splits of $\mathcal{X}$ into smaller and smaller pieces. The fundamental idea is to select each split of a subset so that the data in each of the descendant subsets are "purer" than the data in the parent subset.

11.3.1 The Predictor Space

Suppose that we have p explanatory variables $X_1, \ldots, X_p$ and n observations.

Each of the X_i can be

a) a numeric variable: $\longrightarrow \quad n-1$ possible splits.

b) an ordered factor: $\longrightarrow \quad k-1$ possible splits.

b) an unordered factor: $\longrightarrow \quad 2^{k-1}-1$ possible splits.

We pick the split that results in the greatest decrease in *impurity*. We will soon provide various definitions of impurity.

11.3.2 Deviance as a measure of impurity

A simple approach to classification problems is to assume a multinomial model and then use deviance as a definition of impurity.

Assume $Y \in \mathcal{G} = \{1, 2, \ldots, k\}$.

- At each node i of a classification tree we have a probability distribution p_{ik} over the k classes.
- We observe a random sample n_{ik} from the multinomial distribution specified by the probabilities p_{ik}.
- Given X, the conditional likelihood is then proportional to $\prod_{\text{(leaves } i\text{)}} \prod_{\text{(classes } k\text{)}} p_{ik}^{n_{ik}}$.
- Define a deviance $D = \sum D_i$, where $D_i = -2 \sum_k n_{ik} \log(p_{ik})$.
- Estimate p_{ik} by $\hat{p}_{ik} = \frac{n_{ik}}{n_{i.}}$.

For the olive that we get the following values:

Root	n_{11}	=	246	n_{12}	=	74	n_{13}	=	116	n_1	=	436	D	=	851.2
	$\hat{p}_{11}$	=	$\frac{246}{436}$	$\hat{p}_{12}$	=	$\frac{74}{436}$	$\hat{p}_{13}$	=	$\frac{116}{436}$						
Split 1	n_{11}	=	246	n_{12}	=	0	n_{13}	=	0	n_1	=	246	D	=	254.0
	n_{21}	=	0	n_{22}	=	74	n_{23}	=	116	n_2	=	190			
	$\hat{p}_{11}$	=	1	$\hat{p}_{12}$	=	0	$\hat{p}_{13}$	=	0						
	$\hat{p}_{21}$	=	0	$\hat{p}_{22}$	=	$\frac{74}{190}$	$\hat{p}_{23}$	=	$\frac{116}{190}$						
Split 2	n_{11}	=	246	n_{12}	=	0	n_{13}	=	0	n_1	=	246	D	=	0
	n_{21}	=	0	n_{22}	=	74	n_{23}	=	0	n_2	=	74			
	n_{31}	=	0	n_{32}	=	0	n_{33}	=	116	n_3	=	116			

11.3.3 Other measures of impurity

Other commonly used measures of impurity at a node i in classification trees are

- the entropy: $\sum p_{ik} \log(p_{ik})$.
- the GINI index: $\sum_{j \neq k} p_{ij} p_{ik} = 1 - \sum_k p_{ik}^2$.

For regression trees we use the residual sum of squares:

$$D = \sum_{\text{cases } j} \left(y_j - \mu_{[j]} \right)^2$$

where $\mu_{[j]}$ is the mean of the values in the node that case j belongs to.

11.3.4 Recursive Partitioning

INITIALIZE	All cases in the root node.
REPEAT	Find optimal allowed split.
	Partition leaf according to split.
STOP	Stop when pre-defined criterion is met.

11.4 Model Selection

- Grow a big tree T.
- Consider snipping off terminal subtrees (resulting in so-called rooted subtrees).

- Let R_i be a measure of impurity at leaf i in a tree. Define $R = \sum_i R_i$.
- Define size as the number of leaves in a tree.
- Let $R_\alpha = R + \alpha \times$ size.

The set of rooted subtrees of T that minimize R_α is nested.

11.5 General Points

What's nice:

- Decision trees are very "natural" constructs, in particular when the explanatory variables are categorical (and even better, when they are binary).
- Trees are very easy to explain to non-statisticians.
- The models are invariant under transformations in the predictor space.
- Multi-factor response is easily dealt with.
- The treatment of missing values is more satisfactory than for most other model classes.
- The models go after interactions immediately, rather than as an afterthought.
- The tree growth is actually more efficient than I have described it.

- There are extensions for survival and longitudinal data, and there is an extension called treed models. There is even a Bayesian version of CART.

What's not so nice:

- The tree-space is huge, so we may need a lot of data.
- We might not be able to find the "best" model at all.
- It can be hard to assess uncertainty in inference about trees.
- The results can be quite variable (the tree selection is not very stable).
- Actual additivity becomes a mess in a binary tree.
- Simple trees usually do not have a lot of predictive power.
- There is a selection bias for the splits.

11.6 CART References

- L Breiman.
  Statistical Modeling: The Two Cultures.
  Statistical Science, 16 (3), pp 199-215, 2001.
- L Breiman, JH Friedman, RA Olshen, and CJ Stone.
  Classification and Regression Trees.
  Wadsworth Inc, 1984.

- TM Therneau and EJ Atkinson.
  An Introduction to Recursive Partitioning Using the RPART Routines.
  Technical Report Series No 61, Department of Health Science Research, Mayo Clinic, Rochester, Minnesota, 2000.

- WN Venables and BD Ripley.
  Modern Applied Statistics with S.
  Springer NY, 4th edition, 2002.

11.7 Bagging

- Bagging predictors is a method for generating multiple versions of a predictor and using these to get an aggregated predictor.
- The aggregation averages over the versions when predicting a numerical outcome and does a plurality vote when predicting a class.
- The multiple versions are formed by making bootstrap replicates of the learning set and using these as new learning sets.
- The vital element is the instability of the prediction method. If perturbing the learning set can cause significant changes in the predictor constructed, then bagging can improve accuracy.

Bagging = *B*ootstrap *Agg*regat*ing*

Reference: Breiman L (1996): *Bagging Predictors*, Machine Learning, Vol 24 (2), pp 123-140.

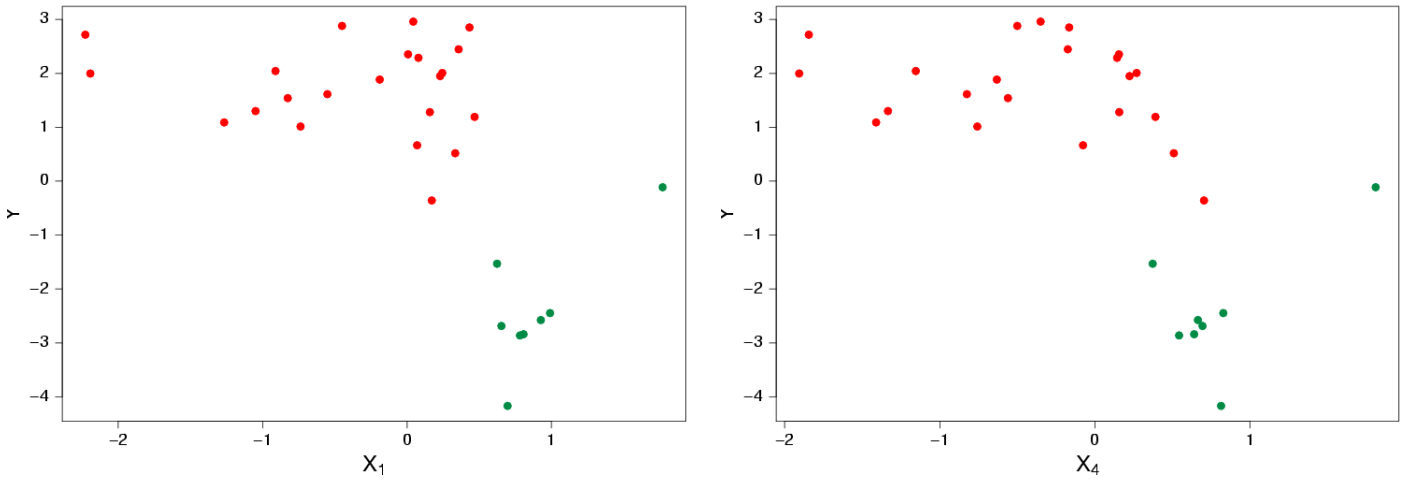


Figure 11.9: Two bagging examples.

- Generate a sample of size $N = 30$ with two classes and $p = 5$ features, each having a standard Gaussian distribution with pairwise correlation 0.95.
- The response was generated as $Y \sim N\left(\mu = 2 - 4 \times I_{[X_1 > 0.5]}\ ,\ \sigma^2 = 1\right)$

A test sample of size 2000 was also generated from the same population.

Note:

- Bagging can dramatically reduce the variance of unstable procedures such as trees, leading to improved prediction.

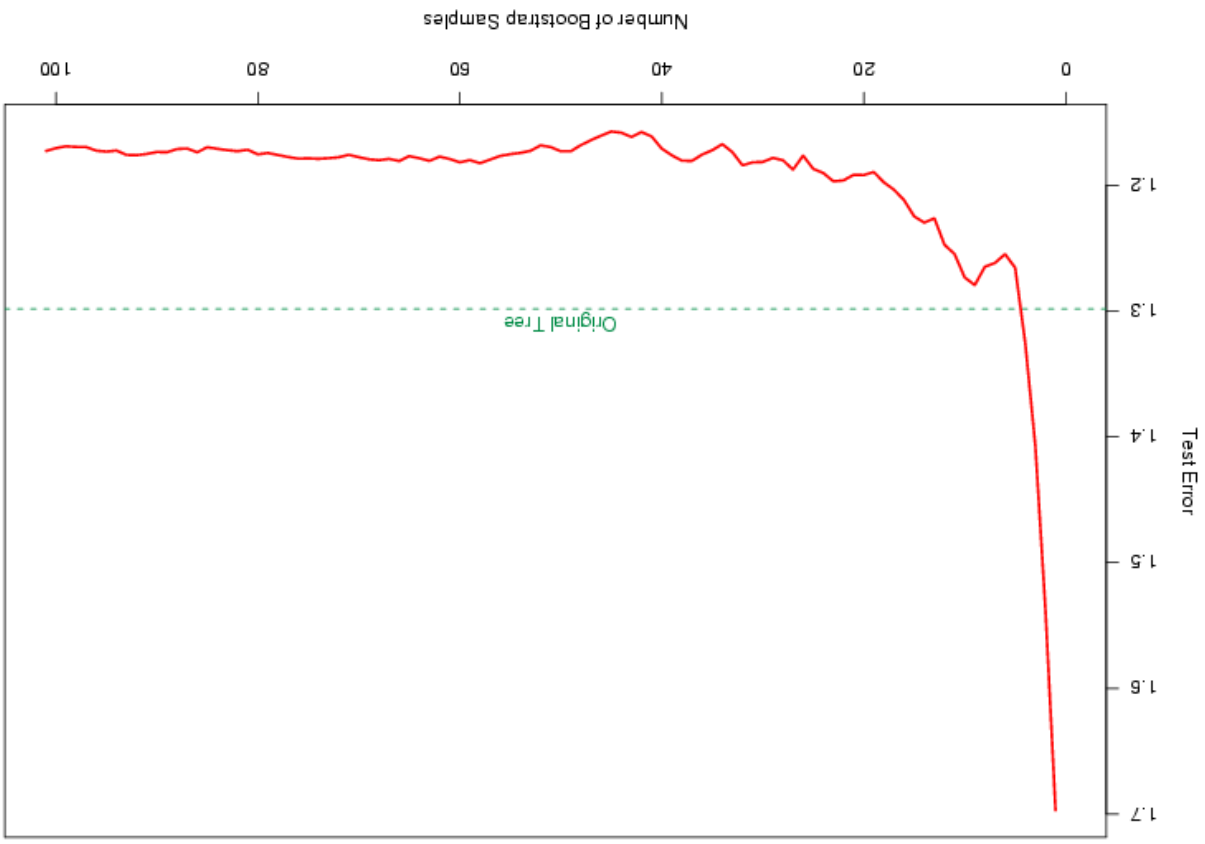


Figure 11.10: Bagging example.

- A simple argument can show why bagging helps under squared error loss: averaging reduces variance and leaves bias unchanged.

Reference: Hastie T, Tibshirani R, and Friedman J (2001): *The Elements of Statistical Learning*, Springer, NY.

However:

- The above argument breaks down for classification under 0-1 loss.
- Other tree-based classifiers such as random split selection perform consistently better.

Reference: Dietterich T (2000): *An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization*, Machine Learning 40:139-157.

11.8 Random Forests

- Grow many classification trees using a probabilistic scheme. $\longrightarrow$ A random forest of trees!
- Classify a new object from an input vector by putting the input vector down each of the trees in the forest.
- Each tree gives a classification (i. e. the tree votes for a class).

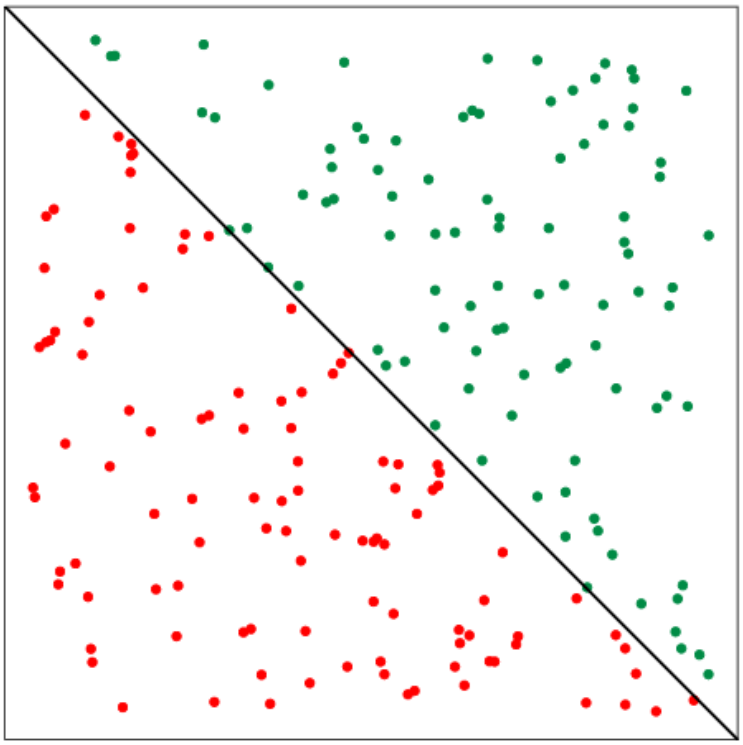


Figure 11.11: boost grid 1

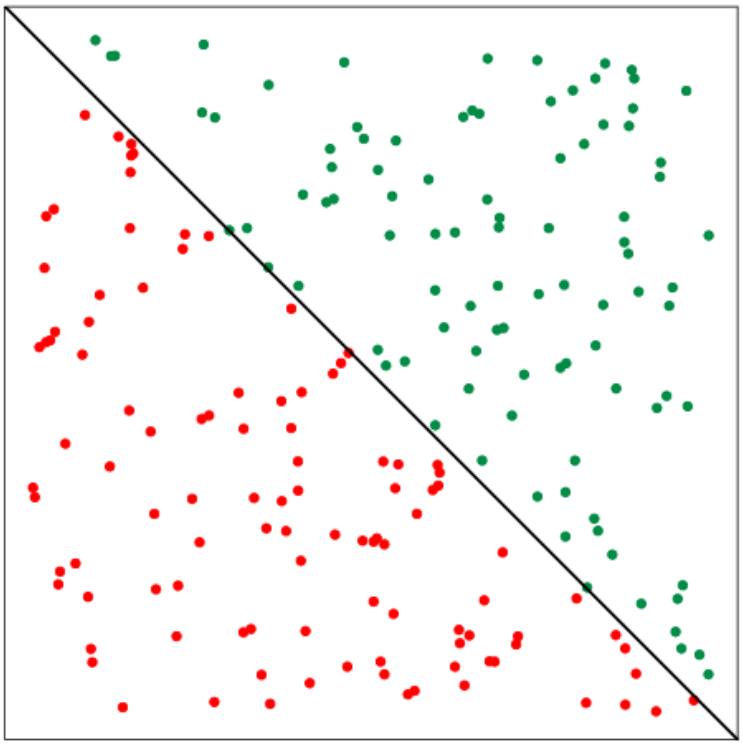


Figure 11.12: bagg.grid.1000.stump

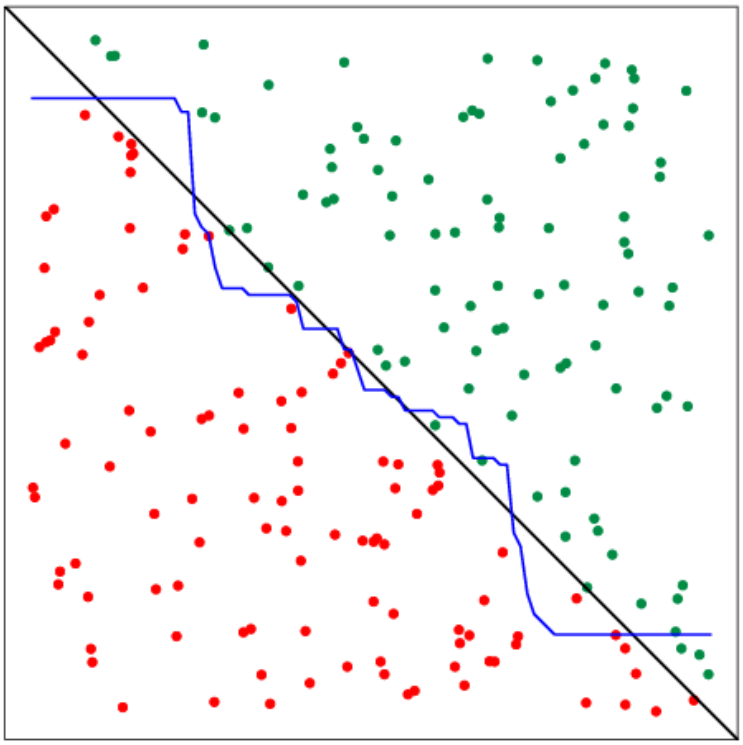


Figure 11.13: bagg.grid.1000.tree

- The forest chooses the classification having the most votes over all the trees in the forest.

Adapted from: *Random Forests* by Leo Breiman and Adele Cutler. `http://www.math.usu.edu/~adel`

Each tree is grown as follows:

1. If the number of cases in the training set is N, sample N cases at random - but with replacement, from the original data. This sample will be the training set for growing the tree.
2. If there are M input variables, a number m $<<$ M is specified such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constant during the forest growing.
3. Each tree is grown to the largest extent possible. There is no pruning.

Two very nice properties of Random Forests:

- You can use the out of bag data to get an unbiased estimate of the classification error.
- It is easy to calculate a measure of "variable importance".

The forest error rate depends on two things:

1. The correlation between any two trees in the forest. Increasing the correlation increases the forest error rate.

2. The strength of each individual tree in the forest. A tree with a low error rate is a strong classifier. Increasing the strength of the individual trees decreases the forest error rate.

$\longrightarrow$ Reducing m reduces both the correlation and the strength. Increasing it increases both. Somewhere in between is an "optimal" range of m - usually quite wide. This is the only adjustable parameter to which random forests is somewhat sensitive.

Reference: Breiman L. *Random Forests*. Machine Learning, 45(1):5-32, 2001.

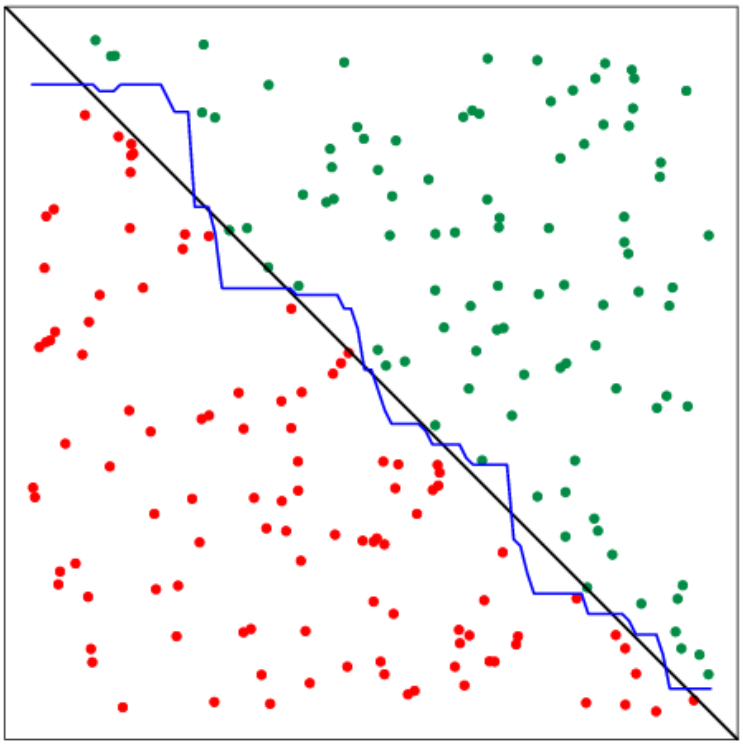


Figure 11.14: Random Forrest rf.grid.tree

11.9 Boosting

Idea: Take a series of weak learners and assemble them into a strong classifier.

Base classifier: $G(X) \to \{-1, +1\}$

Training data: $(x_i, y_i), \ \ i = 1, \ldots, N.$

The most popular version is Adaboost.

$\longrightarrow$

Create a sequence of classifiers, giving higher influence to more accurate classifiers. During the iteration, mis-classified observations get a larger weight in the construction of the next classifier.

Reference: Freund Y and Schapire RE (1996): *Experiments with a New Boosting Algorithm*, Machine Learning: Proceedings of the Thirteenth International Conference, pp 148-156.

1. Initialize the observation weights $w_i = 1/N, \ \ i = 1, \ldots, N.$
2. For $m = 1, \ldots, M$
   (a) Fit a classifier $G_m(x)$ to the training data using the weights w_i.
   (b) Compute
   $$\epsilon_m = \frac{\sum_i w_i \times I_{[y_i \neq G_m(x_i)]}}{\sum_i w_i}.$$

(c) Compute $\alpha_m = \log\left(\frac{1-\epsilon_m}{\epsilon_m}\right)$.

(d) Set $w_i \leftarrow w_i \times \exp\left\{\alpha_m I_{[y_i \neq G_m(x_i)]}\right\}$, $i = 1, \ldots, N$.

3. Output $G(x) = \text{sign}\left[\sum_m \alpha_m G_m(x)\right]$.

- Generate the features $X_1, \ldots, X_{10}$ as standard independent Gaussian.
- The target Y is defined as 1 if $\sum X_j^2 > \chi^2_{10}(0.5)$, and -1 otherwise.
- There are 2000 training cases with approximately 1000 cases in each class, and 10,000 test observations.

11.10 Miscellaneous

- There are many flavors of boosting - even many flavors of Adaboost!
- What we talked about today also goes under the name Arcing: *A*daptive *R*eweighting (or *R*esampling) and *C*ombining.
- There are R packages on CRAN for Random Forests (randomForest) and boosting (gbm).
- Find more details about the issues discussed in Hastie T, Tibshirani R, and Friedman J (2001), *The Elements of Statistical Learning*.

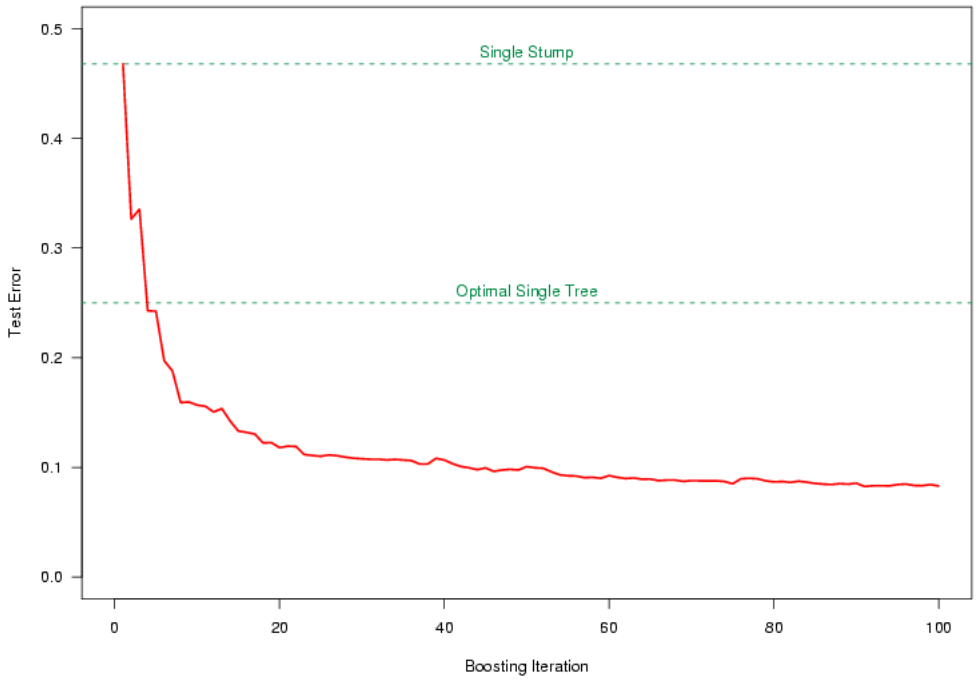


Figure 11.15: Boosting scores

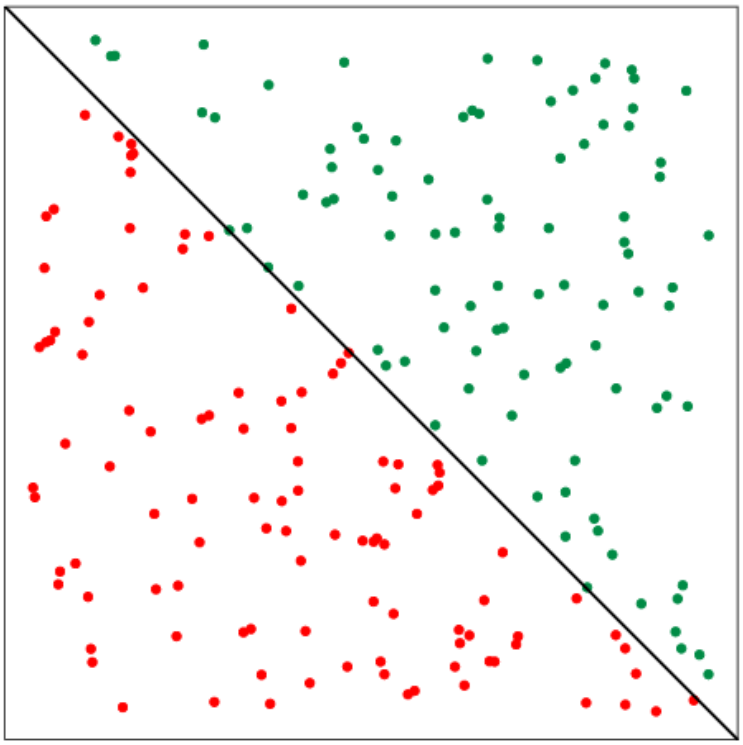


Figure 11.16: Boosting grid 1

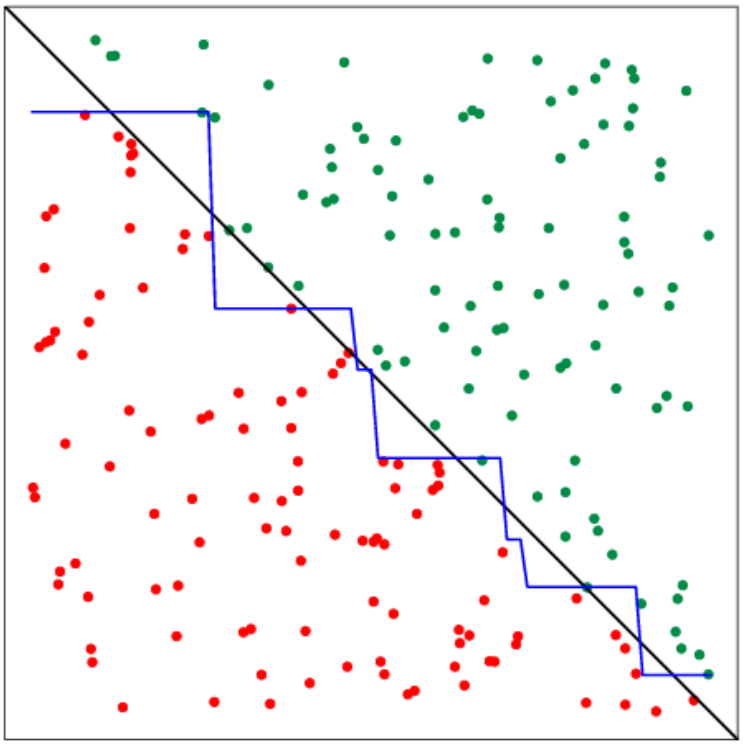


Figure 11.17: Boosting grid 2